Nelson
Reverendo

Navegação autónoma para robôs de serviço em ambientes interiores usando faróis

Indoor autonomous navigation for service robots using beacons

Universidade de Aveiro
2018
Departamento de Eletrónica,
Telecomunicações e Informática

Nelson
Reverendo

Navegação autónoma para robôs de serviço em ambientes interiores usando faróis

Indoor autonomous navigation for service robots using beacons

"*Charge a robot, and it will run as long as its batteries last. Teach a robot how to self-charge and it will run forever!*"

— Nelson Reverendo

Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

Nelson Reverendo

Navegação autónoma para robôs de serviço em ambientes interiores usando faróis

Indoor autonomous navigation for service robots using beacons

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor António José Ribeiro Neves, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor José Manuel Neto Vieira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho à minha noiva Tânia.

o júri / the jury

presidente / president

Prof. Doutor Armando José Formoso de Pinho
Professor Associado C/ Agregação da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Paulo José Cerqueira Gomes da Costa
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto (arguente)

Prof. Doutor António José Ribeiro Neves
Professor Auxiliar da Universidade de Aveiro (orientador)

agradecimentos /
acknowledgements

Ao meu orientador, Doutor António Neves, um especial agradecimento por todo o apoio, paciência e motivação, pois foram um fator decisivo durante o desenvolvimento desta dissertação.
Aos Doutores José Luis Azevedo e José Neto Vieira um obrigado pela ajuda prestada ao nível do hardware.
Ao IEETA, DETI e todos os professores que me acompanharam ao longo destes cinco anos, por terem contribuído para o sucesso do meu percurso académico e garantido todas as condições necessárias para o desenvolvimento deste trabalho.
À minha família, mãe, sogros, avós e cunhada, um especial obrigado por todo o apoio e motivação que me deram nas horas mais difíceis, sem vocês não teria conseguido.
À minha incrível futura esposa, não tenho palavras para agradecer todos os conselhos e todo o apoio nos momentos decisivos deste percurso.

Palavras Chave robótica móvel, navegação autónoma, localização ativa, faróis, ROS.

Resumo

Hoje em dia, os robôs estão cada vez mais presentes no nosso quotidiano, fornecendo uma variedade de serviços e realizando as mais diversas tarefas, algumas delas de forma completamente autónoma.

Para que o robô execute tarefas autónomas deve estar ciente do ambiente que o rodeia e conhecer a sua posição no mesmo. Para atingir esse objetivo, existem três problemas principais a serem resolvidos: mapeamento, localização e navegação.

Durante este trabalho desenvolvemos um robô autónomo de boas-vindas para o Instituto de Engenharia Eletrónica e Informática de Aveiro com a capacidade de receber ordens de um visitante e guiá-lo até ao destino solicitado. No final desta tarefa, o robô retorna autonomamente ao seu local de partida, onde retoma a tarefa de carregamento.

Para atingir este objetivo estudámos algoritmos relacionados com os três problemas referidos. Como exemplo, o algoritmo *GMapping* baseado em laser scans é usado para o processo de Mapeamento e Localização Simultânea, a abordagem adaptativa de localização de Monte Carlo é usada para que o robô que se mova no espaço e o algoritmo *A** é aplicado para planeamento de um caminho.

Foram feitas diversas melhorias em relação ao uso desses algoritmos, incluindo no ambiente um sistema de localização ativa baseado no uso de beacons ultra-som.

O resultado final é um agente autónomo capaz de mapear o edifício, localizar-se no mapa resultante e mover-se da posição atual para um destino especificado. Também é capaz de recalcular o caminho e evitar colisões mínimas em tempo real durante a navegação.

Keywords mobile robotics, autonomous navigation, active localisation, beacons, ROS.

Abstract

Nowadays robots are becoming more present in our daily life performing a variety of on-demand services. In order to perform autonomous tasks the robot should be aware of its environment. To achieve this goal, there are three main problems to solve: mapping, localisation and navigation. During this work, we developed an autonomous welcome robot for the Institute of Electronics and Informatics Engineering of Aveiro (IEETA) with the capacity to receive requests from a visitor and guide him to the requested destination. At the end of this task, the robot should return autonomously to its docking station. To accomplish this goal we studied algorithms related to the three referred problems. As an example, a laser-based solution is used for the Simultaneous Localisation and Mapping procedure (*Gmapping*), the adaptive Monte Carlo localisation approach (AMCL) for the robot moving in 2-D and *A** as a method for path planning. Improvements have been made regarding the use of these algorithms including in the environment an active localisation system based on the use of ultrasound beacons. The end result is an autonomous agent capable of mapping the building, self-localise in the resulting map and moving from current position to a specified target. It is also capable of path recalculation and minimal real-time collision avoidance while navigating.

Contents

List of Figures

List of Tables

Glossary

WLAN	Wireless Local Area Network
GPS	Global Positioning System
D-GPS	Differential-Global Positioning System
QoT	Quality of Trilateration
RSSI	Received Signal Strength Indicator
GSM	Global System for Mobile Communications
TDoA	Time Differences of Arrival
IoT	Internet of Things
DoA	Direction of Arrival
ToA	Time of Arrival
RTT	Round-Trip-Time
IMU	Inertial Measurement Unit
PDR	Pedestrian Dead Reckoning
ROS	Robot Operating System
IEETA	Instituto de Engenharia Electrónica e Telemática de Aveiro
SLAM	Simultaneous Localization and Mapping
LRF	Laser Rangefinder
UV	Unmanned Vehicle
AWR	Autonomous Welcome Robot
HW	Hardware
SW	Software
AMR	Autonomous Mobile Robot
AGV	Autonomous Guided Vehicle
FOR	Field of Regard
GUI	Graphical User Interface
UA	University of Aveiro
KF	Kalman filter
PF	Particle filter
EKF	Extended Kalman filter
UKF	Unscented Kalman filter
MCL	Monte Carlo Localization
AMCL	Adaptive-Monte Carlo Localization
KLD	Kullback-Leibler Distance
RGB-D	Red Green Blue - Depth
IFR	International Federation of Robotics
SERVIR	Sistema de Estações de Referência Virtuais
GNSS	Global Navigation Satellite System
CIGeoE	Centro de Informação Geoespacial do Exército
DGT	Direção-Geral do Território
ReNEP	Rede Nacional de Estações Permanentes
RTABMap	Real-Time Appearance-Based Mapping

CHAPTER 1

Introduction

There has been an increasing demand for mobile robots. From these robots, a variety of tasks is required such as floor cleaning, moving cargo or even performing different assignments along a factory production line. The capability to analyse and understand a large space, when compared to the one observed from a single point of view, is what sets mobile robotics apart from traditional manipulators.

For a robot to achieve complete autonomy, three problems must be solved: mapping, localisation and navigation. The algorithms for each part are well established and documented but, their senseless use is not good practice. In other words, the successive collection of knowledge from the large scale environment must cohesively interact with the self-localisation and real time response parts. This happens because mobile robots are more than a compilation of algorithms for sensing, reasoning and moving, they are a practical application and must cope with all the real world variables [1].

Currently, there are many different solutions on the market. Some take a more familiar form which is the case of *Pepper* (Figure 1.2), a humanoid curator robot capable of recognising human emotion and adjusting its behaviour accordingly, while others such as *Gita* (Figure 1.3), which follows a person while carrying her belongings, and *MiR* (Figure 1.1), a modular robot developed to transport cargo inside a facility, adopt a more peculiar form. Most importantly, human-machine interaction is the common factor in all these solutions.

Figure 1.1: MiR100 (logistics robot) [2].

Figure 1.2: Pepper (humanoid robot) [3].

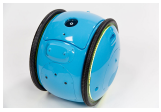


Figure 1.3: Gita (mobile-carrier robot) [4].

In this thesis we propose a solution for the problem of autonomous navigation inside a building. After a study on Kalman filters (KFs) and Particle filters (PFs), the solution for the mapping stage was a PF-based Simultaneous Localization and Mapping (SLAM) technique named *Gmapping*. The chosen algorithm for path planning was $A*$ and for the navigation problem Adaptive-Monte Carlo Localization (AMCL), also PF-based. In this last part (i.e navigation) an improvement to the sole implementation of AMCL was made by adding an active localisation system based on the use of ultrasound beacons. This additional active component enables the system to be as user friendly as possible without the need for a manual starting position setup and also provides robustness to events such as kidnapping.
Two practical applications were developed in this thesis. The first consisted of an agent capable of autonomously mapping a maze and posteriorly navigate to a given location in the maze. This was accomplished using the indoor localisation system.
The second case study consisted of an curator robot for Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA) able to perform autonomous tasks on demand. In this solution, the traditional algorithm for localisation was enhanced with the addition of the ultrasound localisation system.

1.1 Contributions

The main contribution of this thesis is the study of the current alternatives for the different parts that make a functional and reliable autonomous navigation system. The majority of the state-of-the-art algorithms are developed to solve a general problem and, in order to build custom solutions, some configurations and combination of methodologies from distinct fields are required.
With this in mind, the majors contributions of this thesis can be defined as:

- Study the most import important concepts for mobile robotic development and divergent applicational solutions.
- Analyse and review different solutions for indoor tracking systems.
- Review and study the distinct required components to develop a navigation system.
- Development of an user friendly autonomous mobile systems for indoor purposes.
- Study and development of an active beacon system solution to enhance the self-localisation algorithm.

1.2 Thesis structure

This thesis is divided into six chapters:

- Chapter 2 presents a review of the most import aspects in autonomous mobile robotics. Robot Operating System (ROS) is also introduced as core tool for robotic development.

- Chapter 3 is a state-of-the-art analysis of some import methodologies for tracking a mobile device. The focus here is on techniques which have some kind of indoor application. A comparative table is presented at the of this chapter.

- Chapter 4 presents the analysis of the first case study for the thesis, made with *micro-rato*. Here the objective was an autonomous mapping and navigation agent capable of performing task in a maze.

- Chapter 5 is a detailed description of the second case study in which we proposed to develop an Autonomous Welcome Robot (AWR) for IEETA.

- Chapter 6 presents a resumed overview and analysis of the main results and contributions from this work and also indicates some desirable future work.

CHAPTER 2

Autonomous Mobile Robots

Today, robotic applications are no longer restricted to controlled and static environments and must be able to cope with the presence of mutable temporary variables in its surroundings such as people walking by and other robots. The concept of mobile robotics combines the ability of reacting to unforeseen situations while navigating in this uncontrolled environment.
Ranging from the ones performing automated tasks in our homes, up to some moving heavy cargo in industrial facilities, mobile robots come in different shapes and sizes customised to our demand.

To develop a functional mobile robot, several important aspects must be studied. First of all it has to be able to move in the target environment. This means that choosing the appropriate locomotion methodology is a terrain dependent task. Secondly, one must understand the kinematics behind the intended robotic movements. Finally, once the robot is capable of navigating, it has to comprehend the environment's layout. Perception can be seen as the meaningful interpretation of sensory readings.

2.1 Locomotion

Locomotion is the essential mechanism which enables a mobile robot to move freely in its environment. Nowadays, there are a considerable number of possible ways a robot can move. Therefore, the selection of the most appropriate mechanism is a key step in mobile robotics development. Most of these methods are inspired by their biological equivalents (Figure 2.1) due to its success rate when moving through complex environments, as stated in [5].
Currently, many different bipedal robots already perform complex movements such as climbing stairs, jumping, running and even performing back flips. *Atlas* (Figure 2.3) and *ASIMO* (Figure 2.2) are self-balancing two legged robots capable of mimicking the human way of moving. *NAO* (Figure 2.5) is an smaller interactive humanoid robot used in complex human-robot interactions such as helping children with autism.

Also, *AlphaDog* (Figure 2.4), a four legged robot from Boston Dynamics, is a useful tool which can carry heavy loads over rough terrain.
As technology evolved, the most successful and common form was refined and named as wheeled locomotion. In this section we will focus our attention in the wheeled locomotion type as it is the one used for the practical part of this dissertation.

Type of motion	Resistance to motion	Basic kinematics of motion
Flow in a Channel	Hydrodynamic forces	Eddies
Crawl	Friction forces	Longitudinal vibration
Sliding	Friction forces	Transverse vibration
Running	Loss of kinetic energy	Periodic bouncing on a spring
Walking	Loss of kinetic energy	Rolling of a polygon

Figure 2.1: Locomotion mechanisms used in biological systems [5].

Figure 2.2: ASIMO - Honda [6].

Figure 2.3: Atlas - Boston Dynamics [7].

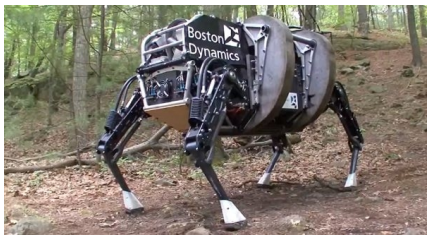


Figure 2.4: AlphaDog – Boston Dynamics [8].

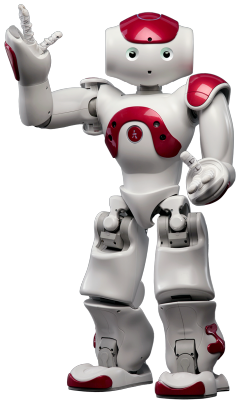


Figure 2.5: NAO - SoftBank Robotics [9].

2.1.1 Wheeled locomotion

Wheeled locomotion solutions are considered the most suitable for common applications. The terrain irregularity is a key restriction. Therefore, the configuration, number and type of wheels (Figure 2.6) used depends on the specific application case. In Figure 2.7 five possible configurations are shown. The first two arrangements (Figures 2.7a and 2.7b), represent differential drive where, in addition to the motorised wheels, one and two extra points of contact are used, respectively. Figure 2.7c represent the configuration for Ackerman steering with two motorised wheels on the back and two steered in the front. Steering has to be different in the front wheels in order to avoid skidding. Also a four wheel configuration, Figure 2.7d demonstrates the use of castor wheels. Finally, Figure 2.7e represents a configuration for an omnidirectional robot using Swedish wheels.

As seen, the minimum number of wheels for stability is two. A differential drive robot can accomplish static stability under certain theoretical conditions, but, practically it usually requires a third point of contact with the ground to handle motor torque. Other robots, can have omnidirectional drive, which means they can move in any direction along the ground plane, at all times.

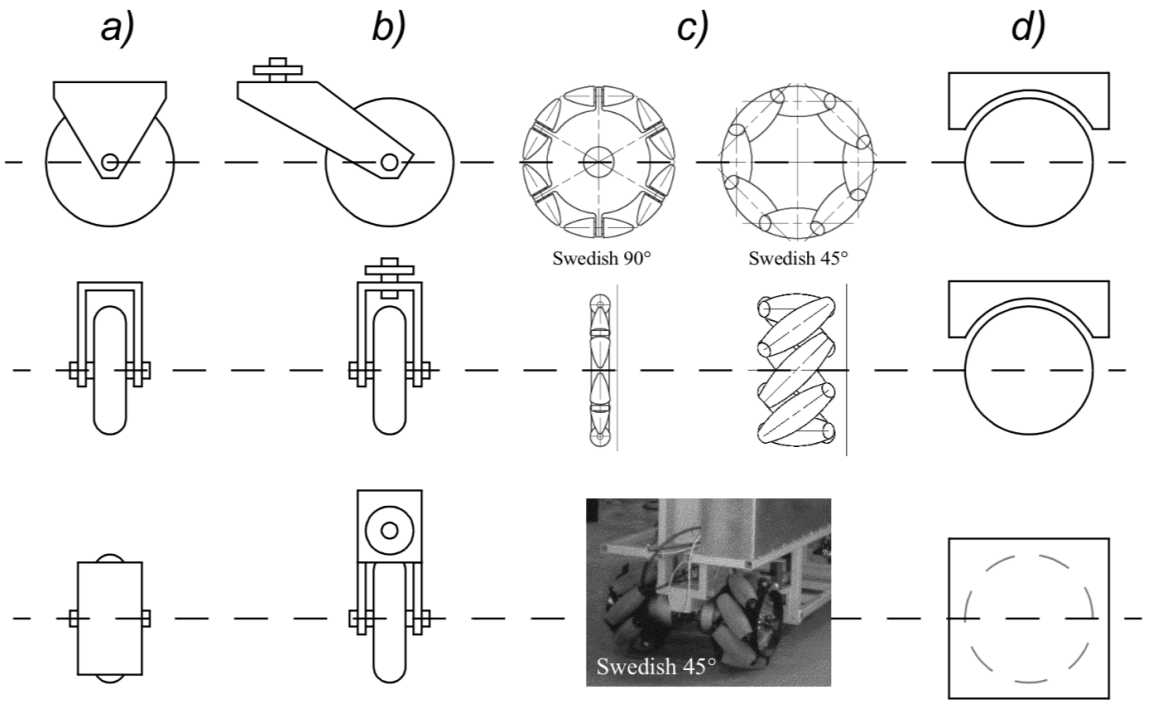
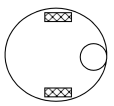
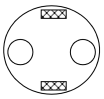


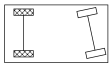
Figure 2.6: The four basic wheel types. (a) Standard wheel: two degrees of freedom; rotation around the (motorised) wheel axle and the contact point.(b) castor wheel: two degrees of freedom; rotation around an offset steering joint. (c) Swedish wheel: three degrees of freedom; rotation around the (motorised) wheel axle, around the rollers, and around the contact point. (d) Ball or spherical wheel: has three hundred sixty degrees of freedom [5].



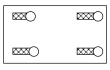
(a) Differential drive with single contact point.



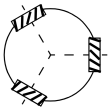
(b) Differential drive with two contact points.



(c) Ackerman steering.



(d) Four motorised castor wheels.



(e) Three Swedish wheels arranged in a triangle.

Figure 2.7: Wheel configuration examples [5].

2.2 Differential drive kinematics

kinematics is the behavioural study of mechanical systems without considering the forces that influence motion. This is an important part of mobile robotic development because it helps to design the best solution for a given task as well as to create the dedicated software for control. Our focus goes to differential drive as it is the one used in both case studies.

In differential drive, wheels are interconnected with the robot's chassis geometry and their joined constraints form the global constraints for the overall robot movement. Forces and constraints must be represented accordingly to a uniform reference frame, especially for mobile robotics in which the trajectory is defined by the speed of each wheel. A mapping between local and global reference frame is demonstrated in Figure 2.8. Considering the robot as a point (P), its position in the global reference frame can be stated as a three element vector

(Equation 2.1) containing the coordinates x and y and the angular difference between both reference frames (θ).

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.1}$$

Further detailed formulas and mathematical expressions to relate global and local frames can be found in [5].

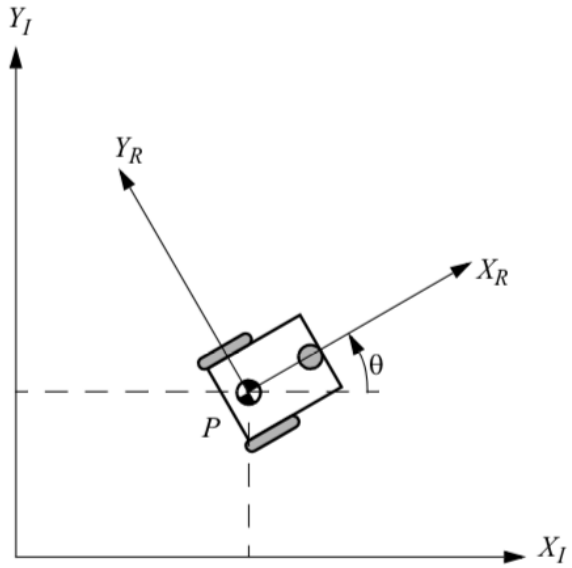


Figure 2.8: Mapping between global reference frame $\{X_I, Y_I\}$ and the robot local reference frame $\{X_R, Y_R\}$ [5].

2.3 Perception

In order to perceive its environment, any autonomous system has to convert measurements from a variety of sensors into useful information. Sensor can be divided in two functional categories: *proprioceptive/exteroceptive* and *passive/active*. *Proprioceptive* sensors read values internal to the system such as motors speed and battery status. On the other hand, *exteroceptive* sensors measure information related to the robot's environment like distance, temperature and sound. These measurements must then be interpreted as stated before.

Active sensors are the ones emitting some form of energy into the robot's surroundings and measuring its reaction (e.g radars, lasers, ultrasound, etc). Usually they achieve higher rates of accuracy, but may also suffer from interference (e.g ultrasound interference) from other sensors or from external factors. As opposite, *passive* sensors read values of energy entering the sensor (e.g temperature, cameras, microphones, etc).

In form of example, cameras (Figure 2.10) are passive sensors which capture the light coming through the lens to form an image (Figure 2.12). On the other hand, Laser Rangefinders (LRFs) are active sensor that emit a laser beam into the medium.

Safety is a major concern in automated systems, and for this reason many solutions use

ultrasonic and Infrared sensors for collision avoidance.
Sensors are also used for Mapping. In this case, Stereo and Red Green Blue - Depth (RGB-D) cameras are commonly used to build depth maps. It is also import to reference that LRFs was the mapping sensor used in the second practical case study.
Everett in [10] presents detailed information about some traditional sensors used in mobile robotics.

2.3.1 LRF

LRFs (Figure 2.9) are exteroceptive sensors used for many applications, starting with simple distance measurements and going up to the ones used for mobile robotics. Compared with other sensors, LRFs provide dense and accurate range measurements with a high sampling rate and angular resolution. A LRF uses a laser beam to compute the distance to an obstacle. The most common forms of LRFs are based on the time of flight method which consists of sending a pulse towards the object and measuring the time taken until it is reflected on it and received by the sender. Nguyen, Martinelli, Tomatis, *et al.* in [11] provide a comparison between some Line Extraction Algorithms using 2-D LRF indoors. SLAM algorithms such as Gmapping use data readings (Figure 2.11) from these lasers to created a usable map, as the studies from [12] and [13] show.

Figure 2.9: SICK LMS - LRF [14].

Figure 2.10: Onboard Point Grey camera.

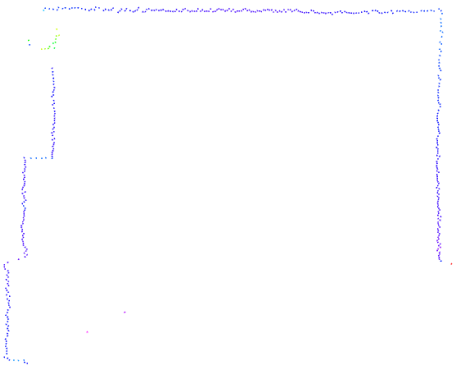


Figure 2.11: Raw reading from a LRF.

Figure 2.12: Raw image from the onboard Point Grey camera.

2.4 Mobile robotic applications

When discussing mobile robotics, two major differentiations can be made. The first, designated as Autonomous Mobile Robot (AMR) (Figure 2.13), where robots move and navigate uncontrolled environments without the use of guidance devices. This type, usually has an elevated computational capacity which allows the robot to self-locate in a pre-constructed map of its environment, therefore being able to compute the shortest path to a destination and, if an unmapped obstacle emerges, act accordingly (i.e can determine another path to destination or maneuver around it).
In contrast, Autonomous Guided Vehicles (AGVs) (Figure 2.14) rely on guidance devices to navigate from one place to the other, following a pre-defined path, in a considerably controlled environment. With reduced computational power when compared with AMRs, AGVs are controlled by simpler programming instructions and require additional Hardware (HW) components such as wires and magnetic strips. These extra elements imply, extensive and expensive facility changes during installation time or if new routes are necessary. Opposing the already described way for obstacle avoidance of AMRs, as it can't calculate a new path, an AGV, when faced with a blockage in its trajectory, simply stops until the path is cleared.

Figure 2.13: Turtlebot-2 as an AMR example [15].

Figure 2.14: Industrial truck as an AGV example [16].

2.5 Service robots

As stated by the International Federation of Robotics (IFR) [17], a service robot is an agent "that performs useful tasks for humans or equipment excluding industrial automation applications". In other words, service robots assist human beings with dangerous, repetitive and dirty jobs. It is such an appealing area, that an international competition from *RoboCup*[1] was created. In this challenge named *@Home*, the focal point is the development of home assistance and care robots. Besides the ones already presented in Chapter 1, other solutions are in place such as *AMIGO* a service and care taking robot, *Roomba* an automated vacuum

[1] `http://www.robocup.org`

cleaner and *PatrolBot*, a general purpose robot which can perform tasks like environment monitoring, autonomous deliveries and even security checks.

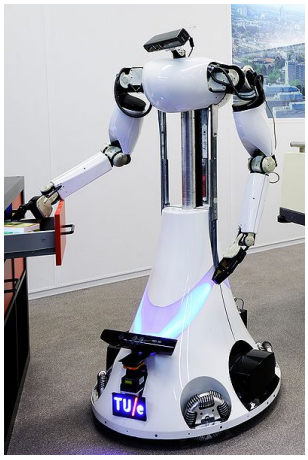


Figure 2.15: AMIGO-service and care taking robot [18].

Figure 2.16: ROOMBA-autonomous vacuum cleaner [19].

Figure 2.17: PatrolBot-autonomous security robot [20].

2.6 ROS

Today, developing software for robotic applications is a complex task due to the variety of available hardware and the enormous size of the required code. ROS, a robot software framework, emerged to overcome these issues and to turn code reuse into a simpler operation. In spite of not being an operating system, ROS provides hardware abstraction for heterogeneous computer systems.

ROS philosophy comprehends three major aspects [21]:

1. *Peer to peer*: it consists of a group of small interconnected programs (nodes) that regularly exchange messages to specific topics (Figure 2.18). Without the use of a central routing service, the result is an easily scalable system.

2. *Tools-based*: it does not have an authoritative development and runtime environment. Different programs perform the distinct task such as visualising system interconnection and generating documentation. This strongly encourages the development of new and improved implementations for specific domains.

3. *Multilingual*: choosing a specific programming language can be a question of taste or a necessity. Having this in mind, ROS users can program modules in any of the different languages with already developed client libraries (e.g Python, C++, Java, etc). These client libraries communicate among them selves using a predefined convention for message serialisation.

A complete description of ROS is presented in [21] and [22].

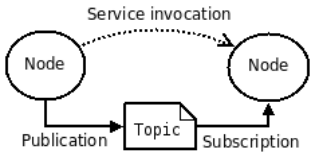


Figure 2.18: ROS basic node communication concept [23].

CHAPTER 3

Location-Based Technologies

Mobile device tracking is the act of finding a device's location. The methods and technology used may vary depending on the objective (indoor/outdoor) and accuracy necessary.
Outdoor tracking refers to all areas without roof and/or walls as opposed to indoor which involves all building topologies.
In this chapter, we aim to study the different state-of-the-art techniques and technologies available to track a mobile device's absolute position, compare the accuracy/reliability and setup cost of each method. These approaches were developed with different goals and we will study them, when possible, taking into account the dissertation's objective as opposed to a global analysis. Also, the ones developed for indoor purposes are emphasised and analysed to a higher degree of detail.

In order to execute its tasks, an autonomous robot must know its location. Today, a variety of location based technologies are already in place for the most different reasons. This technology can be used, by itself, to localise a robotic agent or a mobile device attached to it as in the first case study of this thesis. Moreover, it can also serve as complement to the complex localisation algorithms. This is seen in our second case study.
In this chapter, evidence for the importance of choosing the right technology and methodologies for locating a mobile device, is provided. A knowledge base is developed to give us the starting point for implementing our specific localisation system.

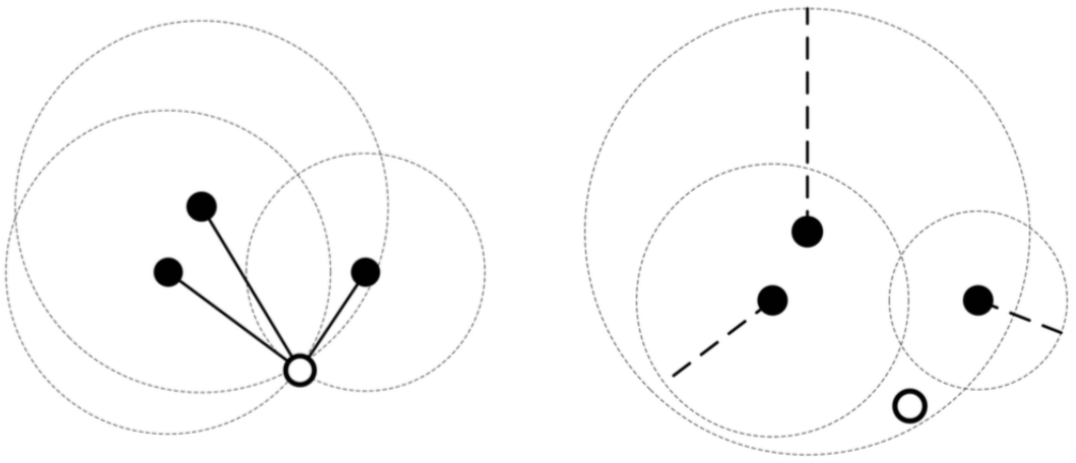
3.1 Multilateration and RSSI

Multilateration is a process used to determine an object's position using range measurements from, at least, three location known beacons as seen in Figure 3.1a. The object's position is considered unique as long as the three beacons are nonlinear.
Multilateration faces many challenges and can have large error due to its channel noise sensitiveness as displayed in Figure 3.1b. To evaluate Multilateration exactness, Yang and

Liu in [24] propose a quantifier designated by Quality of Trilateration (QoT) that quantifies the geometric relationship of objects and ranging noises (i.e different geometric trilateration forms provide distinct accuracy levels) and, therefore, helping to make trilateration choices. Nowadays, context-aware systems are becoming a trend and, to implement them, the necessity to use the technology already in place emerges. This tech is around us, in the case of wireless signals used to defuse the Internet signal through the facilities where we live/work, or in the palm of our hands like gadgets (e.g mobile phone, tablets, computers, etc) used by us and which possess Bluetooth, Wireless Local Area Network (WLAN) and many different sensors at our disposal.

As there is no direct way to measure distances from signals, such as Bluetooth and WLAN. The solution lays on the use of various signal parameters, as the Received Signal Strength Indicator (RSSI) [25]. The resulting distance is then used in the Multilateration process. A complete analysis of Bluetooth signal parameters is provided in [26].

Also, Bekkelien in [25] provides a comparison in terms of accuracy between WLAN and Bluetooth for indoor positioning systems.



(a) Ideal trilateration result. **(b)** Noisy trilateration result.

Figure 3.1: Ideal versus Noisy multilateration using three beacons (trilateration) [24].

3.1.1 RSSI distance determination

RSSI is based on the assumption that transmission power from de transmitter device (P_{TX}) affects the receiving power at the receiver device (P_{RX}). The received signal strength decreases quadratically with the distance to transmitter device (Figure 3.2) according to Friis' free space transmission equation (3.1):

$$P_{RX} = P_{TX} \cdot G_{TX} \cdot G_{RX} \cdot \left(\frac{\lambda}{4\pi d}\right)^2 \tag{3.1}$$

where P_{TX} = Transmission power of sender , P_{RX} = Power of wave at receiver, G_{RX} = Gain of receiver, G_{TX} = Gain of transmitter, λ = Wave length, d = distance from sender to receiver.

In real world applications, the direct use of this method, for itself, do not guarantee accurate results due to the signal interference from different sources (e.g electromagnetic fields, reflections on metallic surfaces, etc) [27].

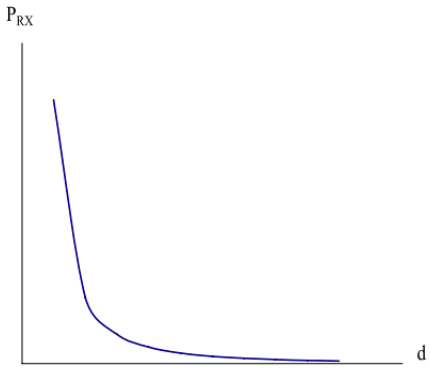


Figure 3.2: Received power vs distance to transmitter [27].

3.1.2 Fingerprinting for accuracy improvement

When tracking devices using Wireless technologies, the location is based on RSSI measurements of the environment's beacons at the receiver device. An estimate pose is supported on the knowledge from signal propagation inside a building. This propagation model can be obtained from a simulation or using calibration measurements at strategic points of the building. *Fingerprinting* is the name given to a collection of these calibration points. A detailed explanation of *Fingerprinting* stages is provided in Section 3.4.2. As stated in [28], which presents a detailed analysis of *Fingerprinting* influence on RSSI-based location systems, re-calibration is needed if a considerable change is made in the environment (e.g access point relocated, furniture displaced, etc). It also reads, that the use of *Fingerprinting* results in an improved and scalable indoor location system. Bekkelien in [25] refers to a 45% average error decrease using the combination of *Multilateration* with *Fingerprinting*.

3.2 GPS vs DGPS

Global Positioning System (GPS) is a Space-based navigation system that provides geolocation and time to a GPS receiver such as a low cost mobile phone [29].
The location is calculated using *Multilateration*, as described in [30], which determines the absolute pose of a point by measurement of distances to, at least, four satellites. Up, in Space, a twenty-four satellite array orbits in six different twelve-hour paths to provide a minimum of five satellite coverage over every point on the planet as shown in Figure 3.3.
As an improvement to standard GPS, Differential-Global Positioning System (D-GPS) was developed, illustrated in Figure 3.4. The use of two relatively close receivers (Fixed Base Stations **(R1)** and Rover **(R2)**), when compared with the distance travelled by space travelling signals, enables a computation of the difference between estimated and actual signal travel time. This process relies on the assumption of equal signal error received in **R1** and **R2**. The

referenced computation is made for all visible satellites by **R1** and posteriorly broadcasted to **R2** which applies it to the ones it is currently using.
Also a Global Navigation Satellite System (GNSS), Sistema de Estações de Referência Virtuais (SERVIR) network was developed by the Portuguese Army and consists of a set of permanent reference stations (Figure 3.5) for GNSS observation. It covers continental Portugal and provides real-time positioning corrections as well as post-processing data. Centro de Informação Geoespacial do Exército (CIGeoE) claim a three-dimensional (X, Y, Z) accuracy better than five centimetres (5 cm).
Rede Nacional de Estações Permanentes (ReNEP) (Figure 3.6) a similar Portuguese system provided by Direção-Geral do Território (DGT), supplies ten centimetre (10 cm) accuracy geographic localisation data to GPS users.

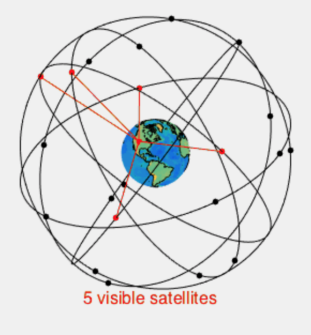


Figure 3.3: Array of 24 satellites revolving around earth (5 visible at current time) [31].

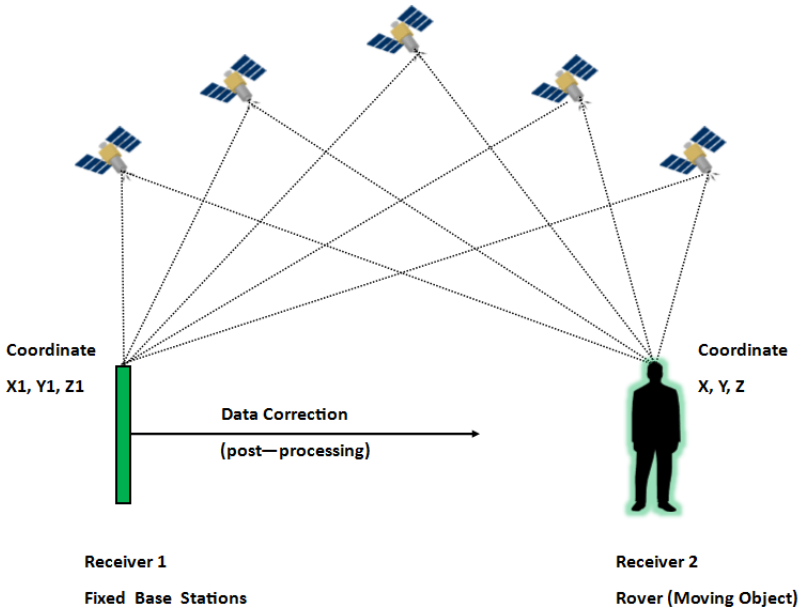


Figure 3.4: D-GPS working principle [32].

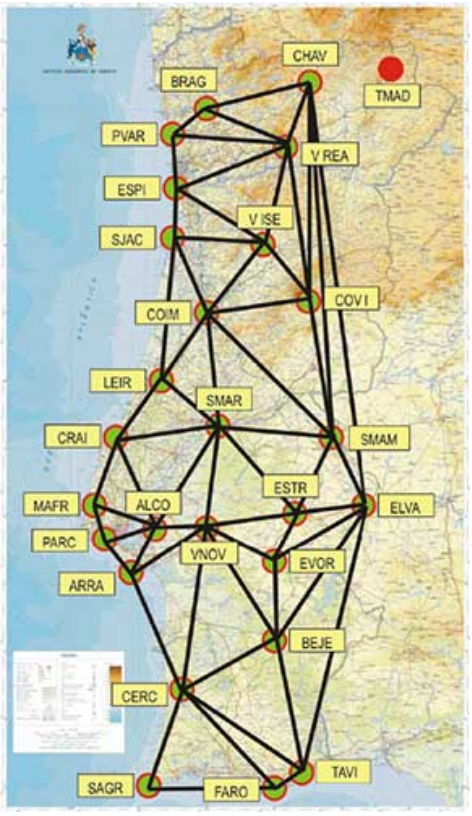


Figure 3.5: SERVIR reference stations distribuition [33].

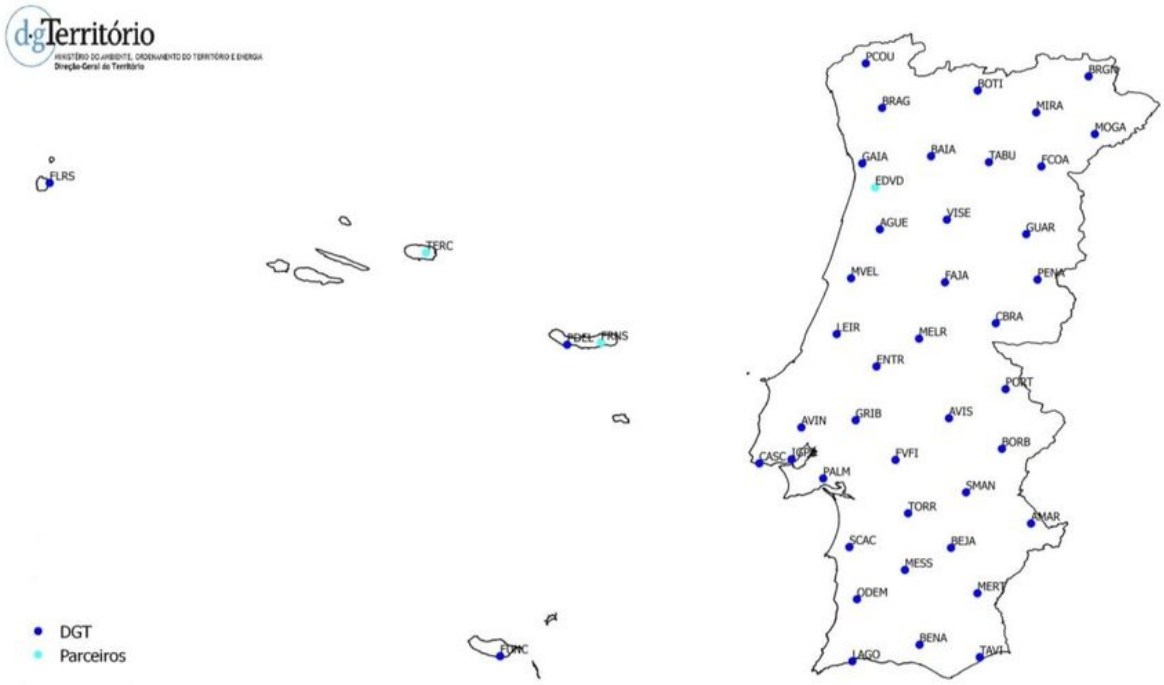


Figure 3.6: ReNEP permanent stations distribuition [34].

3.3 GSM

Global System for Mobile Communications (GSM) is a standard developed to describe the protocols for second-generation digital cellular networks used by mobile devices. Usually, GSM base stations are equipped with a number of directional antennas that define sectors of coverage or cells as seen in Figure 3.7. Being a cellular network means devices connect to it by searching for cells in the immediate vicinity. There are different cell sizes which are mainly placed in outdoor environments.
Some work was done using GSM for accurate mobile device tracking [35]–[37] in which the six-strongest cells (GSM standard) were combined with up to twenty-nine other weak cells (i.e signal too low to be used in communications but strong enough to be detected). This approach is highly dependent on a good *Fingerprinting* which is time consuming, and is building topology dependent. Also, as Denby, Oussar, Ahriz, *et al.* [37] reads on its conclusion that, for better accuracy, the full GSM carrier set must be taken into account, due to the variation of coverage being carrier dependent.

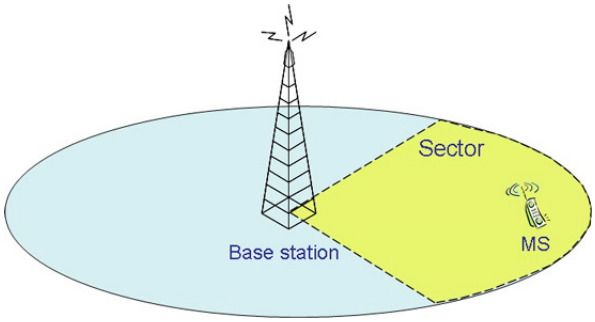


Figure 3.7: GSM sector division [38].

3.4 WLAN

Internet of Things (IoT) devices presence in day-to-day life is growing exponentially and, with it, WLAN infrastructures presence in almost every building of the developed world. Therefore, when going for a WLAN tracking solution, the system installation costs are reduced and this enables an accuracy oriented budget. There are two main variation of WLAN solutions, one without and other with the use of *Fingerprinting* techniques.

3.4.1 Without *Fingerprinting*

Research using *Smart Antennas* was conducted in [39]. As explained in detail in [40], [41], *Smart Antennas* consist of antenna arrays which make use of smart signal processing algorithms to identify spatial signal signatures such as the signal's Direction of Arrival (DoA), and use them to calculate *Beamforming* vectors which enables antenna beam tracking/locating on a mobile device.
The supra referenced research, concludes that it outperforms other conventional techniques without the need of a training phase or database. The solution presented in [42], a software - based technique using Time of Arrival (ToA) obtained from Round-Trip-Time (RTT) combined

with Kalman filter [43], obtains accuracies up to one meter. Also, the simple use of RSSI conversion to distance and posterior application of *Multilateration*, provided good accuracy after the signal strength to distance conversion has been calibrated for the area [44].

3.4.2 With *Fingerprinting*

As reviewed in [45], some methodologies involve the use of *Fingerprinting* from RSSI measurements. This variation can be divided in two stages, one *online* and other *offline*.

Offline phase

In this phase, also called calibration phase, a collection of various RSSI readings from strategic calibration points is obtained. Many factors, such as device orientation, temperature, humidity, furniture displacements and human beings mobility, can influence this method, resulting in a highly complex process of acquiring data. Furthermore, the quality of this phase is very time dependent (i.e depends on when the calibration data was obtained).

Online phase

In this, sometimes designated as online tracking, phase, the real time RSSI value is compared with the ones obtained in the previous phase and, the position corresponding to the most similar *Fingerprint* is chosen as the estimated device position.

3.5 Accoustic Signals

In recent years, some studies on device tracking using accoustic signals have been made. Two particular variations are in place, one in which the mobile device acts as the emitter **(E)** [46] and other where it uses the incorporated microphone to receive sound **(R)** [47]. These accoustic signal have an identifier code modulated on the signal and are, mainly, beyond the audible range.
In the first variation, **E**, the mobile phone emits high pitched accoustic signals which are captured by receivers placed in strategic points of the building. For instance, in [46] the position of mobile device is calculated by *Multilateration* of Time Differences of Arrival (TDoA), using specific algorithms.
As for the second case, **R**, an interesting research was carried out for video taping drones [47], where the drone emits accoustic sound that are captured by a mobile phone, which computes the relative position of the drone and sends, back to the drone, control commands in order to keep it following the mobile phone.

3.6 Ultrasound beacons

Nowadays, beacons are used for a variety of context aware applications. Several types of beacons are on the market using different technologies like bluetooth, Infrared and ultrasound. Ultrasound beacons are active devices, emitting timed signals. The advantage of having an active beacon is that, encoded in the signal, we can send information relative to the beacon (e.g ID, battery level, etc) therefore making it easier to distinguish each device. Typically, beacon

localisation relies on a group of beacons placed in known locations along the environment. Some studies were performed using ultrasound beacon for localisation [48], [49].

3.6.1 Marvelmind Indoor Navigation System

In the practical part of this thesis we used an indoor navigation system from Marvelmind. It's designed with robotic systems in mind which makes system integration quite simple.
The navigation is based on stationary ultrasonic beacons communicating in a license-free band [50]. To compute the location of the mobile beacon, trilateration is applied to the propagation delay of the ultrasonic signal in a set of stationary beacons.
An advantage of the system is the automated map build feature (i.e no coordinate entry and no manual distance measurement). Each beacon is equipped with five ultrasonic sensors and each sensor has ninety degrees of ultrasonic coverage as seen in Figure 3.9. To communicate and synchronise a proprietary protocol is used. A beacon configured as stationary, must be mounted on walls or ceilings above the robot and with a sensor facing down. To provide maximum ultrasonic coverage in the upper hemisphere, a mobile beacon should be placed horizontally on the robot with its sensor facing up.
To control the system, the modem (Figure 3.8) must be powered on at all times and within radio coverage of all beacons. This component is also used to configure and update the system with the Dashboard interface (Figure 3.10).

Figure 3.8: Marvelmind modem/router [50].

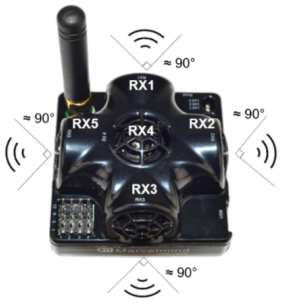


Figure 3.9: Marvelmind beacon ultrasonic coverage [50].

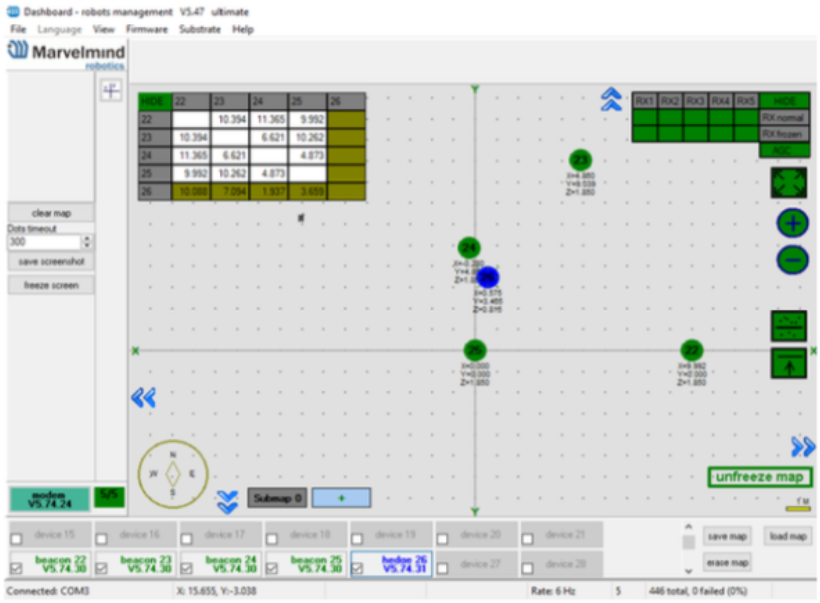


Figure 3.10: Marvelmind Dashboard interface [50].

3.7 Sensor Fusion

Sensor fusion can be seen as the combined use of data coming from multiple sensors in order to improve the localisation accuracy.

As an example, the single use of techniques like Pedestrian Dead Reckoning (PDR), which is the process of estimating a person's current position using information from the past position and/or estimated speed over elapsed time and course, or WLAN has limitations (e.g WLAN signal variations, PDR drift) and may not be a reliable method for tracking. Therefore, novel approaches fuse the best of both worlds, systems already in place like WLAN and state-of-the-art mobile devices with various integrated sensors.

Chen, Zou, Jiang, *et al.* [51] presents a sensor fusion solution combining WLAN, PDR and specific sensor patterns in the environment, designated landmarks, in a system with one meter accuracy. It, also, contains a comparison between their approach and individual sensor approaches.

Moreover, other approaches use Inertial Measurement Units (IMUs), as reported in [52], but instead of building it and spending countless hours calibrating the system, this approach uses the already calibrated sensors inside smart phones to provide all the information needed and resulting in a system with less than point three meters (0.3 m) accuracy.

Table 3.1 presents a comparison of the studied approaches.

Technique	Pros	Cons
GPS	• low cost receiver • easy to implement	• reduced indoor coverage • low accuracy for indoor purposes
Accoustic Signals	• potential high accuracy • widely available hardware (speakers and microphones)	• sensitive to microphone's orientation • user movement speed affects performance
GSM	• no additional interfaces • no building energy source • no frequency interference (commercial dedicated frequency)	• complex fingerprinting (high granularity grid) • build insulation affects accuracy
WLAN w/o Fingerprinting	• low computational cost (over w/ fingerprinting)	• environment dependent
WLAN w Fingerprinting	• improved accuracy (over w/o fingerprinting)	• complex fingerprinting • high computational cost • environment dependent
Fusion	• improved accuracy • minimal calibration • low infrastructural changes	• state-of-the-art device required

Table 3.1: Comparison of mobile device tracking techniques.

CHAPTER 4

Autonomous Navigation with Micro-Mouse

Micro-Mouse (pt: Micro-Rato) is a student competition which occurs in the University of Aveiro (UA) for educational purposes. In this competition the mouse has to solve a maze and find a beacon named cheese. Figure 4.1 shows the competition taking place in the UA's pavilion. A robot with small dimensions is typically used.

The ability to autonomously navigate a maze can be considered as a multidisciplinary task. Primarily, the agent must use some strategy to explore and interpret its environment. As soon it has a usable map, it should localise itself in it. And finally, it can perform autonomous navigation tasks. To develop this case study we used a robot from the UA, traditionally used in the competition referenced before. This robot (Figure 4.2) has the HW configuration seen in Figure 4.3 and is controlled by a *PIC32* micro-controller.
The proposed challenge consists of the development of a mapping and navigation module for the *Micro-Rato* using beacons. We intended to create a robotic agent that maps it's environment. Later, the robot should be able to localise itself in the map and navigate to a given point, starting at any other point.

Figure 4.1: Micro-Rato competition [53].

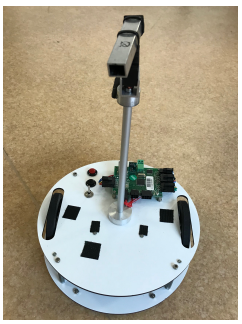


Figure 4.2: Micro-Rato robot.

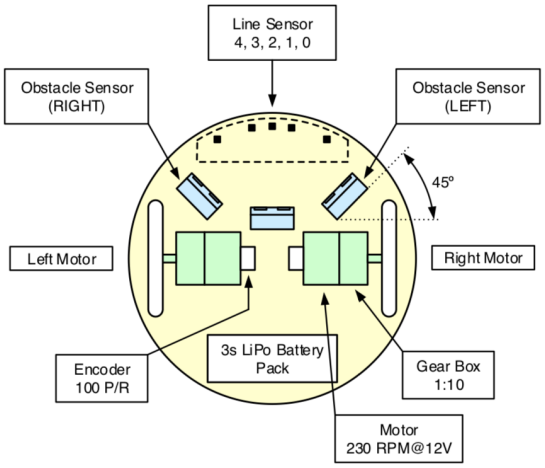


Figure 4.3: Micro-Rato HW specifications [54].

4.1 Path Planning

Path planning consists of computing a path for a robotic agent in a way that it can reach the desired goal without colliding with known obstacles. Often, the robot is considered as a point in space therefore, obstacles must be inflated proportionally to compensate. Usually, the map is transformed into a graph and the path planning problem converted to the shortest path problem (graph theory).

The shortest path problem is the task of finding a path between two nodes in a graph such that the weight sum of its constituent edges is minimised [55] as represented in Figure 4.4. A variety of algorithms emerged to solve this problem, among them *Dijkstra's* algorithm and $A*$ search algorithm which are two of the most well-known.

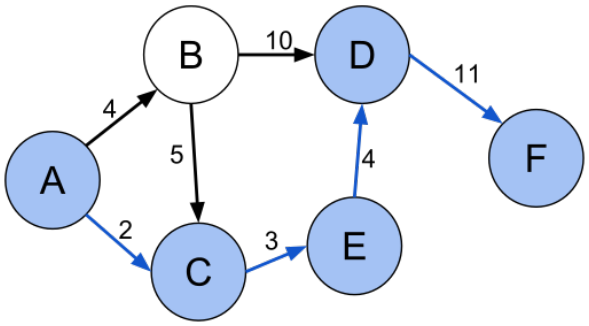


Figure 4.4: Shortest path (A, C, E, D, F) between vertices A and F in the weighted graph [56].

4.1.1 A*

$A*$ is a *best-first* search algorithm which finds the path by searching for the one with the smallest cost (i.e distance, time, etc), through all possible paths to the goal. Starting from a given node of the graph, a set of paths is constructed by expanding each path one step at a time, until one reaches the intended target node. Figure 4.5 shows the path calculated by $A*$

for a robotic agent. The idea is to rank the options, choose the best one, and try it. Options are evaluated according to the cost function (Equation 4.1):

$$f^*(n) = g(n) + h^*(n) \tag{4.1}$$

where '*' means it's an estimate, $g(n)$ represents the cost of going from the starting node to node n, $h^*(n)$ is an estimated cost of moving from node n to the goal.
Function h^* is an heuristic function which is a form of 'guessing' the cost of going from n to the goal node and cannot be higher than the real cost (h) (i.e cost must be underestimated). The best way might be to represent the straight-line distance (i.e Euclidean distance) to the goal, since that is physically the smallest possible distance between any two points.

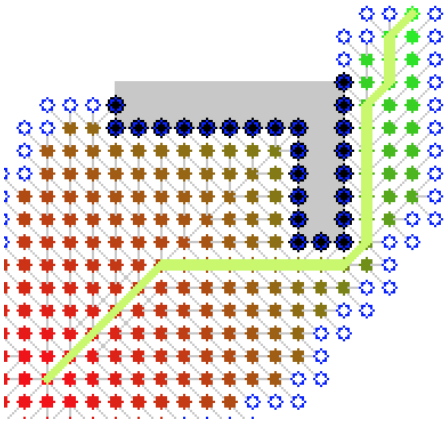


Figure 4.5: *A** search for finding path from a start node to a goal node in a robot motion planning problem [57].

4.2 Architecture

The architecture of this case study can be divided in three major components. The first designated as Exploration phase, in which a map of the maze is obtained, a second component, Mapping and *A**, where the obtained data is converted to usable input for the shortest path algorithm and said algorithm is executed. Finally, the agent moves from a starting point to a given destination on the map, following the path determined by the *A**.
The localisation is made with a set of ultrasound beacons, configured having one beacon near each corner of the maze, and two beacons on the robot, as seen in Figures 4.7 and 4.8 respectively.
For added computational power, a Raspberry Pi was mounted on top of Micro-Mouse establishing a serial Communication with the micro-controller. Figure 4.6 represents the different component interactions. The Raspberry Pi is connected to one of the mobile beacons which provides the positions of both mobile beacons. On the other hand, the *PIC32* is responsible for controlling the motors and for reading the infra-red obstacle sensors.

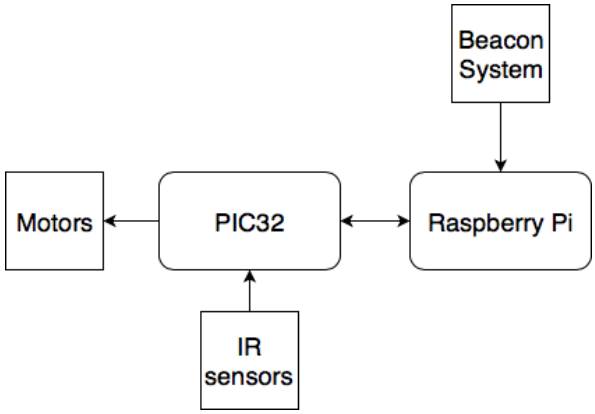


Figure 4.6: Diagram representing the system components interaction.

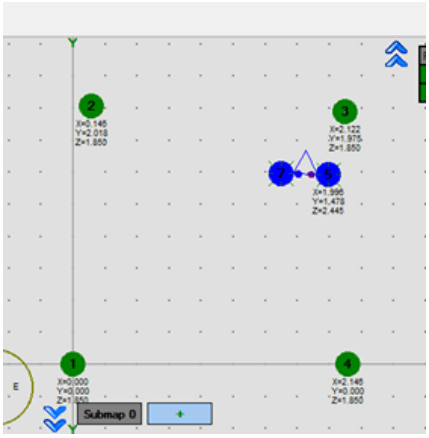


Figure 4.7: Beacon configuration as shown by proprietary software.

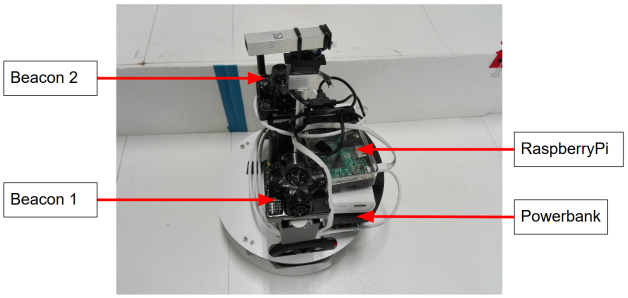
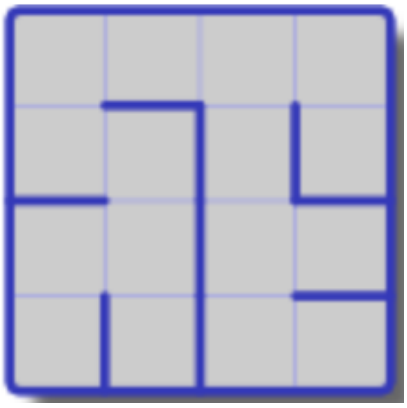


Figure 4.8: Robot configuration.

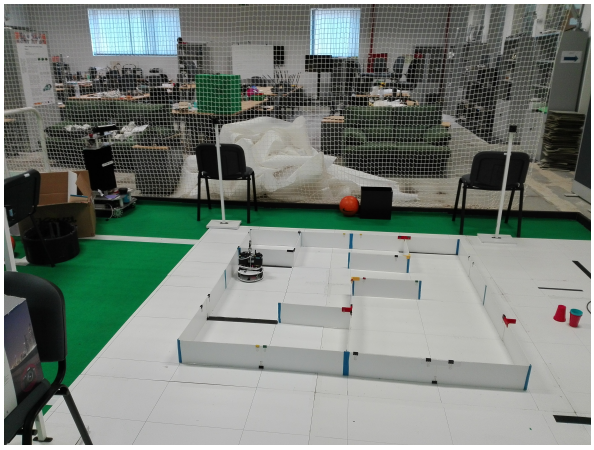
4.2.1 Exploration Phase

During the exploration phase, a wall following algorithm was developed. Using it, the robot travels along the maze following walls, as close as possible, on it's right side. This was possible by using the robot's obstacle sensors. It continually reads the sensor values and adjusts the velocity of the wheels to ensure that it remains within a certain distance of the wall (i.e if the distance increases, the right wheel's velocity decreases and vice versa). When the front sensor detects an obstacle a ninety degree (90°) rotation to the left is performed.

As the robot moves along the wall, a set of points, registered by the robot's right beacon (the closest to the wall), is stored in a file for later use in the mapping. It is important to reference that this process is completely autonomous. The robot can be started in any part of the maze, as long as it is close to a wall, and once it reaches the starting position (i.e has completed a full lap around the maze) it stops the exploration phase. The maze (Figure 4.9) must not have separated walls (e.g islands) for the algorithm to work.



(a) Grid view of an example maze.



(b) Example of a real maze used.

Figure 4.9: Example of *Micro-Rato* mazes.

4.2.2 Mapping and A*

Given a list of points obtained during Exploration phase, the coordinates of those points are scaled down and transformed into integers from 0 to 11, forming a 12 by 12 occupancy grid. This coordinate transformation can be seen in Code 1.

Note that *xmax*, *xmin*, *ymax* and *ymin* variables represent the highest and lowest readings from each coordinate x and y respectively. These values are used to compensate the difference between beacon created Field of Regard (FOR) and actual map space (i.e the map is inside the FOR but smaller). The d variable is the distance from registered points to the wall, and can be easily calibrated.

After this process, with this 12 by 12 matrix, we can define a list of adjacent points forming the maze walls. The points not in this list make the list of adjacencies which serve as input for the $A*$ algorithm which computes the shortest path given the start and finish cells.

```
def transf(self):
        a = []          # x
        b = []          # y
        c = []          # cells
        for x, y, t in visited:
                a.append(int(((x-xmin+d) * 12) / (xmax - xmin + 2 * d)))
                b.append(int(((y-ymin+d) * 12) / (ymax - ymin + 2 * d)))
        for x, y in zip(a, b):
                if (x, y) not in c: # remove duplicated
                        c.append((x, y))
        return c
```

Code 1: Transformation function used to convert points into 12x12 grid cells.

4.2.3 Error Reduction

A good navigation system relies on precise mapping techniques. Therefore, to reduce some encountered errors during the mapping stage, two filters were developed. They can be applied together or separate according to the specific list of resulting points.

Distance Filter

When a the robot rotates to complete a turn, a small error from the beacon system can cause crucial cells to be considered as occupied as seen in Figure 4.10. In order to reduce this, a filter that removes points with a distance below a certain threshold is applied to the list of visited points. The result is presented in Figure 4.11.

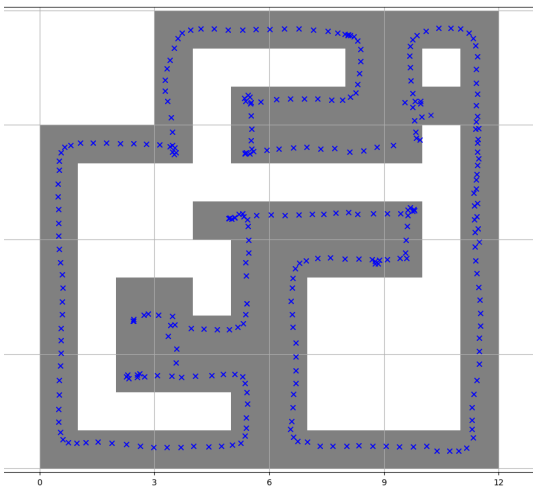


Figure 4.10: Map before distance filter.

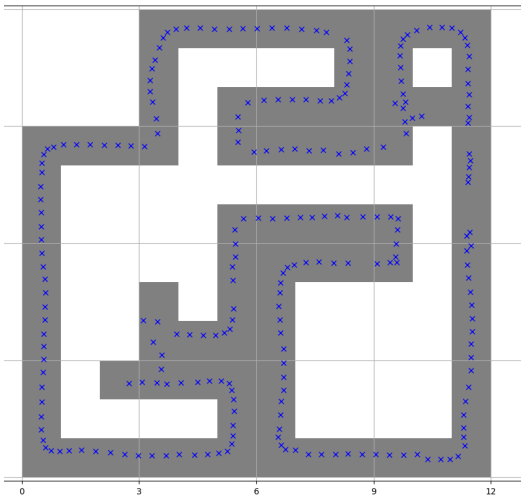


Figure 4.11: Map after distance filter.

Isolated Cell Filter

The beacon system can cause a random point to be marked in a random cell, therefore causing it to be considered occupied. Also, after the Distance Filter is applied, cells with a single point may exist.
This filter prevents, in both cases, the robot considering these cells as occupied. Here, the false negative (i.e cells wrongfully marked as empty) consideration is preferred to the false positive. The result of the applied filter can be seen in Figure 4.12 and in comparison with the real world environment (Figure 4.13) we can state that a good mapping was accomplished.

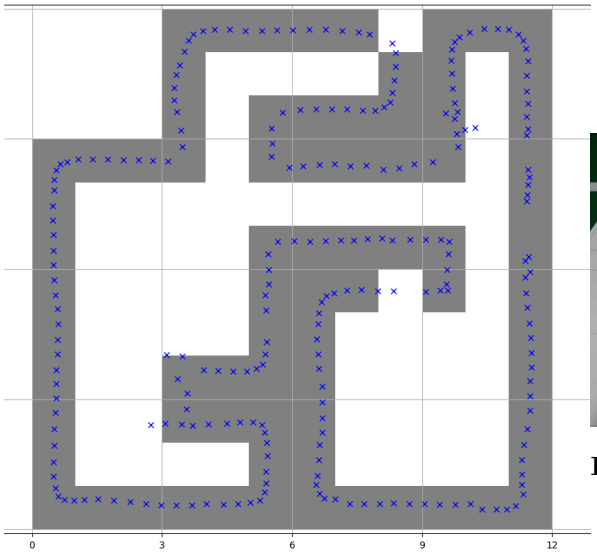


Figure 4.12: Map when isolated cell filter is applied.

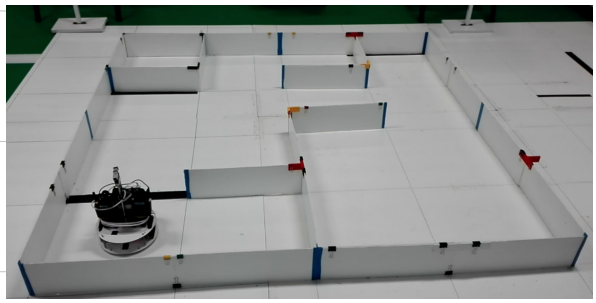


Figure 4.13: Image from the maze used during development.

4.2.4 Returning Phase

After the $A*$ algorithm is executed, it returns a list of cells corresponding to the path to be followed by the robot. The first element in the list is analysed and converted to a set of robot commands (i.e move, rotate, etc.). Once reached, it is removed from the list and the following cell goes under the same process until the list is empty and the target cell is reached.

4.3 Topology Limitations

In order to have a good system performance, each beacon must not have a distance to its neighbours higher than thirty meters. When aiming for 2-D tracking, an unobstructed ultrasound line of sight of two or more stationary beacons is required at all times, for each mobile beacon. Marvelmind states that the beacon system has a precision of two centimeters ($\pm$ 2cm), in its manufacturing data-sheet. Practical precision tests were performed as presented in Section 4.6.

4.4 System Limitations

While developing this case study some difficulties emerged. Firstly, the beacon software available in the manufacturer's web-page, introduces a random delay in the pose readings which could not be completely removed. This delay can be the consequence of bad system configuration or some kind of sonar interference.
Other major hurdle in the beginning, was to establish a serial connection between the microcontroller and the Raspberry Pi. The solution for this problem was obtained by using an adaptation of a C++ module to open the serial communication. Also, in the middle of the development process there were disconnection problems in the Raspberry Pi hardware, during

runtime. It was later discovered that the cable that was supplying energy to the Raspberry Pi was faulty and wasn't giving the necessary input for the hardware to function correctly.

4.5 Complementary Software

During the development of the project some tasks were automated using python scripts. The most important one was the upload of the source code to the Raspberry Pi and download of the mapping data to the user's computer. This was achieved by establishing an SSH connection with the Raspberry Pi.
The usual terminal interface is available and allows control and output visualization of all the system functions. A debug option is also available to test different system behaviours.
To reduce the complexity of the system interaction and to enable an easier data visualization, a Graphical User Interface (GUI) was developed, using a python GUI library named Kivy [1]. Figure 4.14 presents a screenshot from said GUI.

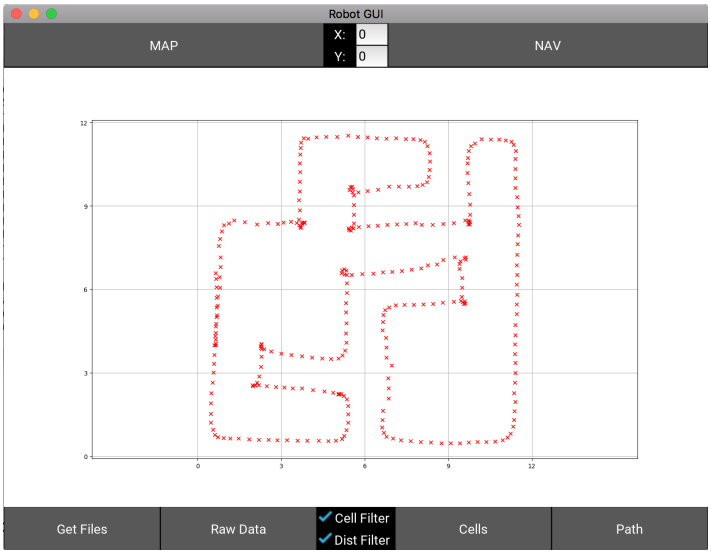


Figure 4.14: GUI screenshot.

4.6 Results

To verify the beacon system's accuracy, tests were made comparing the measures given by the system and the real measurements for reference. Four beacons were arranged in a square, of variable sides in the arrangement seen in Figure 4.15, and a fifth beacon was located exactly at the center of the square, Figure 4.16. To improve the correctness of the physical setup two beacons on one side were aligned with the border of the field in the lab, and the angles were maintained with wooden boards. Table 4.1 present the result of the performed measurements.

[1] `https://kivy.org`

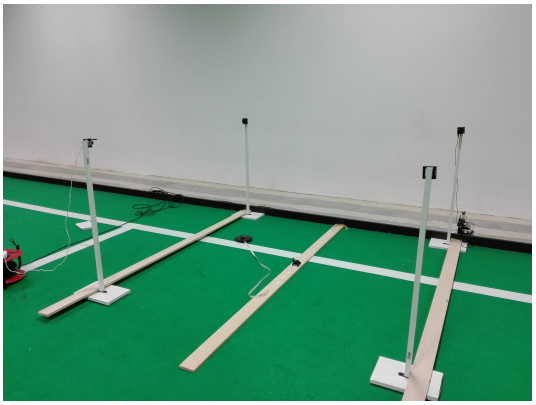


Figure 4.15: Overview of the measurement setup.

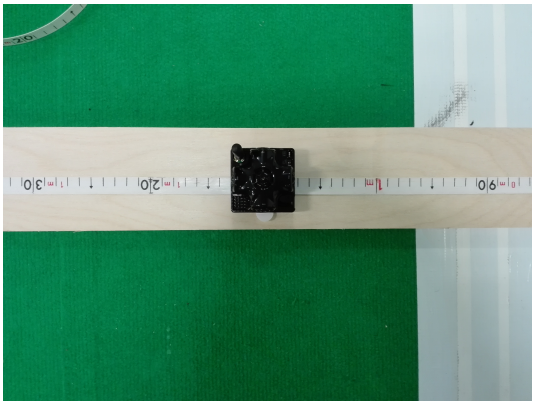


Figure 4.16: Centre beacon of the setup.

Square side (m)	Real coordinates (m)	Measured coordinates (m)
1	(0.5, 0.5)	(0.57 ,0.46)
2	(1, 1)	(1.0, 0.99)
4	(2, 2)	(1.95 ,2.01)
8	(4, 4)	(4.07 ,3.98)

Table 4.1: Real world measurement tests with the beacon system.

In order to compute an overall mapping accuracy, we used the image from Figure 4.17 which contains the beacon marked positions (red crosses) and the wall representation (black lines). In this case, three factors contributed to the mapping error. The first was the beacon system, followed by the distance tolerance allowed by the wall following algorithm. Lastly, the Infra-red obstacle sensors also induced mapping error.
Applying Distance transform from OpenCV[2] we obtain Figure 4.18 in which the pixel grey-level intensity represents the distance to the closest boundary. This gives us a matrix where we can obtain the distance values for each of the marked points during the mapping stage.
Knowing that the robot follows the wall at a distance of eight centimetres (8 cm) and that it corresponds to twenty-eight pixels (28 px) in Figure 4.17, we can estimate a global mapping error. The average value for the marked points is forty-one pixels (41 px) and subtracting the wall following distance we obtain thirteen pixels (13 px) which represents an error close to four centimeters (4 cm).

[2] `https://opencv.org`

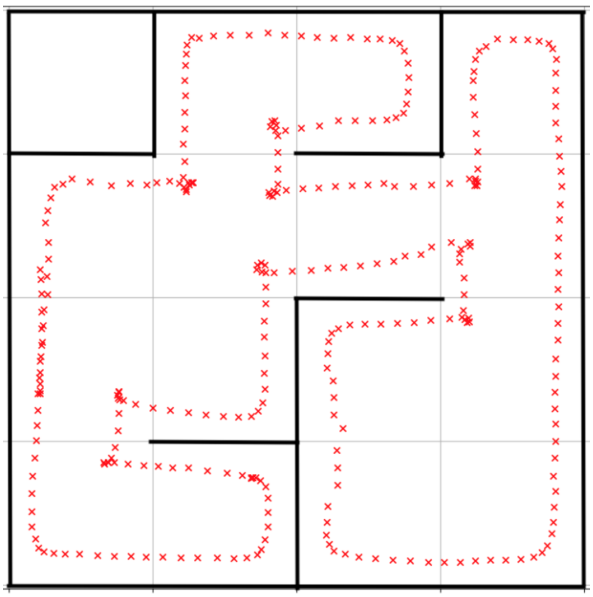


Figure 4.17: Maze's wall representation (black lines) and beacon marked positions (red crosses) for the mapping stage.

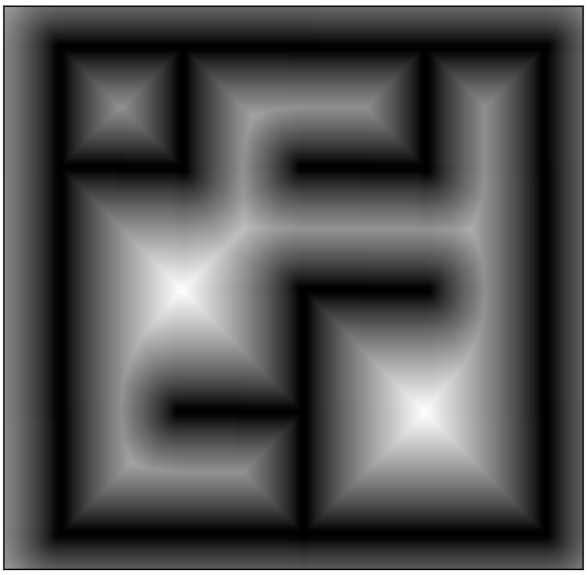


Figure 4.18: Distance Transform result where pixel grey-level intensity represents the distance to closest boundary.

CHAPTER 5

Autonomous Welcome Robot for IEETA

The main case study of this thesis was the development of an Autonomous Welcome Robot (AWR) for Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA) in which the goal is to have an agent capable of performing autonomous tasks in the building environment. From a global point of view, it will have the capability of receiving requests from a visitor and guide him to the requested destination. At the end of this task, the robot should return autonomously to its docking station where it can charge its batteries and await further instructions.

The ability for self-localisation in an environment can be considered one of the cornerstones of mobile robotic applications development. Any good navigation system relies on a precise position estimation which is important in tasks such as map building and path planning.
From a top perspective, we can divide a navigation system in three sections: mapping, path planning and navigation. Mapping is the phase where a static map of the environment is built. Path planning is the section in which the path from a start point to a goal position is computed, taking the known static obstacles into account and finally the navigation step where the actual movement happens and in which the dynamic component of the environment is also considered.

5.1 SLAM

SLAM is the process of build or updating a map of an unknown environment while, simultaneously, tracking the mapping agent's pose. SLAM has many applications, ranging from Unmanned Vehicle (UV) (e.g water, aerial) and self-driving cars to smaller indoor robots, therefore, different solutions emerged to fulfil this diversity of applications using a variety of sensors (e.g 2-D, 3-D, RGB-D).

Extensive research has been done when using SLAM for autonomous vehicles, proving it is possible for an agent to start in an unknown location and environment and, gradually, generate a map of its surroundings [58].
Currently, there are many SLAM algorithms depending on the type of environment and goal, such as online SLAM [59], [60] which is used for dynamic environments. Also, Sturm, Engelhard, Endres, *et al.* in [61] present some work done with RGB-D SLAM and [62], [63] comparisons between some techniques used for mobile robots.
All state-of-the-art algorithms rely on probabilities due to the ability to represent measurement and estimation uncertainty and reliability in the measurement noise, as [64] reads. Furthermore, to solve the mapping problem, many of these probabilistic solutions, rely on Bayes rule [65].
Being a popular Bayesian implementation [66], Kalman filter (KF) is a two step methodology. A prior *prediction* step for estimating the current state variables and their uncertainties. When a new measurement from the sensors is available, the previous prediction is *updated* using a weighted average, with the weight being proportional to the estimation certainty. This recursive method runs in real time using only the most recent sensor output and the last predicted state combined with respective uncertainty matrix (i.e no extra past information). Figure 5.1 represents KF working principle.

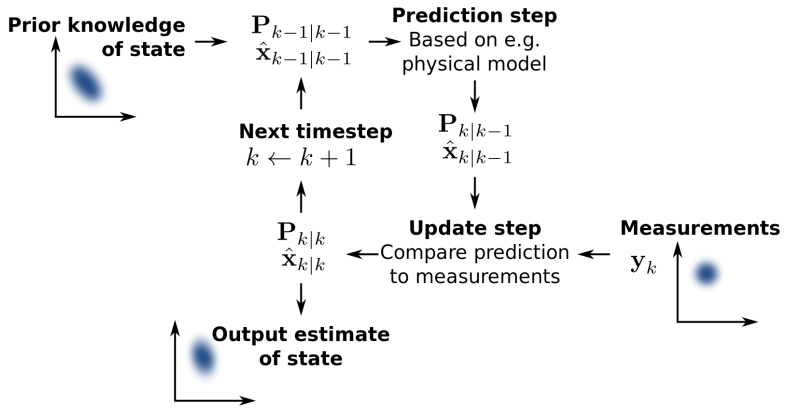


Figure 5.1: KF working principle where the estimate is updated using a state transition model and the measurements. $\hat{X}_{k|k-1}$ corresponds to the estimated state at step k and $P_{k|k-1}$ is the respective uncertainty. [67].

Other KF derived approaches, such as Extended Kalman filter (EKF) [68] and Unscented Kalman filter (UKF) [69], emerged to solve the non-linearity problem in the robot pose model. Particle filter (PF) [70] is another application of Bayes filter where a set of particles is used and associated with a weight which indicates the quality of that specific particle. The estimate of the variable of interest is obtained by the weighted sum of all particles. PF similarly to KF uses the two step process. At the end of each action, the particles are modified in concordance to the model and noise is added to emulate the noise effect on the variable of interest, this makes the *prediction* phase. At the *update* phase, the particle's weight is readjusted based on

the most recent measurements from the sensor.
Other relevant algorithm is graph-based SLAM [71] which is referenced in the literature [72] as being able to fill some of PF and respective derivative algorithms weaknesses. This approach replaces sensor readings and motion events by edges in a graph which can be visualised as "virtual measurements" and represent constraints between consecutive poses. The map is generated throw the linearisation of all the constraints which generates a sparse matrix (i.e majority of elements are equal to zero) representing the graph. A map obtained with a graph-based technique can be seen in Figure 5.2.

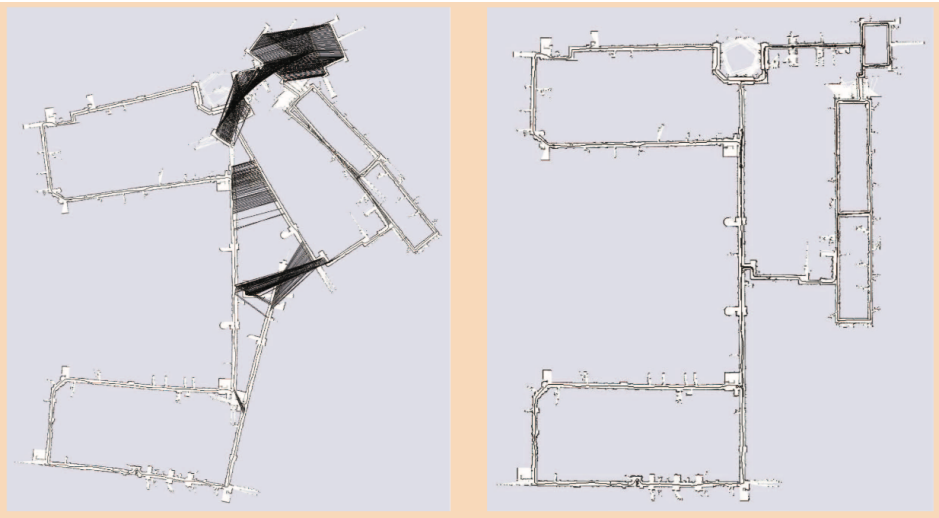


Figure 5.2: Pose-graph corresponding to a data-set recorded at MIT Killian Court. The image on the left corresponds to the map before optimization and the one on the right to a post optimization map.The maps are obtained by rendering the laser scans according to the robot positions in the graph [71].

5.1.1 GMapping

Being one of the most used SLAM packages in robotic agents, particularly the ones using ROS, GMapping [73] is a Rao-Blackwellized PF SLAM based algorithm proposed byGrisetti, Stachniss, and Burgard.
As described in [74] each particle contains an individual environment map leading to an elevated number of particles problem and increased computational complexity. This problem is tackled by the adaptive techniques for grid mapping present in Rao-Blackwellized PF. Such process uses both robot movement and most recent sensor reading to decrease pose uncertainty in the *prediction* phase. Selective re-sampling is also implemented to diminish the particle depletion problem (i.e reduced number of particles) which could reduce the accuracy considerably. Figure 5.3 shows a map from Freiburg Campus obtained with GMapping.

Figure 5.3: Map of the Freiburg Campus obtained with GMapping [74].

5.1.2 Other Approaches

As referenced before some approaches are based on graphs which is the case of Cartographer and RTABMap. Cartographer [75] provides real-time SLAM organised in submaps and with a variety of sensor configurations. Its scans are stored in a probability grid containing the probability of a cell being free.

Moreover, Hector SLAM [76] was developed for UV and handheld mapping devices. It is not graph-based and does not contain any global optimisation technique. Scan matching is performed using a Gauss-Newton equation to best fit the laser beams with the map. Once accepted, the new sensor data is written directly on the grid map.

Nüchter, Bleier, Schauer, *et al.* in [77] state that indoor environments with several features are no challenge for these algorithms. Problems emerge when they are faced with featureless environments such as long tunnels. Figure 5.4 shows an erroneous map obtained with Cartographer and with Hector SLAM in the same place.

Also graph-based, Real-Time Appearance-Based Mapping (RTABMap) [78] is an appearance-based approach for loop closure detection. Using visual features from RGB images, recognises previously visited locations and builds a dense map from point clouds (Figure 5.5). This is useful, when combined with laser readings, to tackle the kidnapped robot problem.

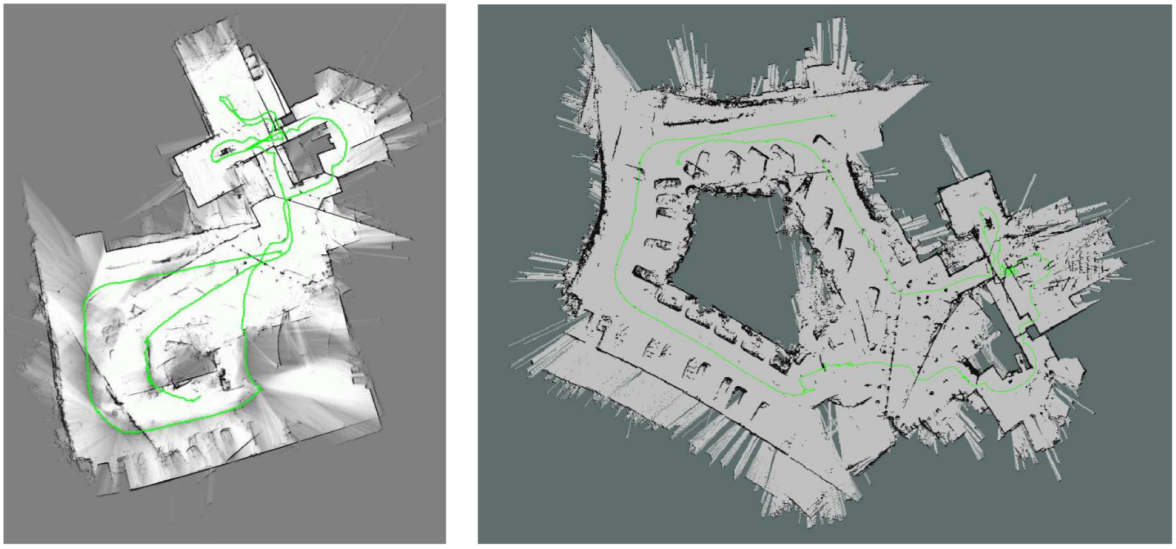


Figure 5.4: Erroneous map obtained with Cartographer (left) vs Hector SLAM (right) in the same location [77].

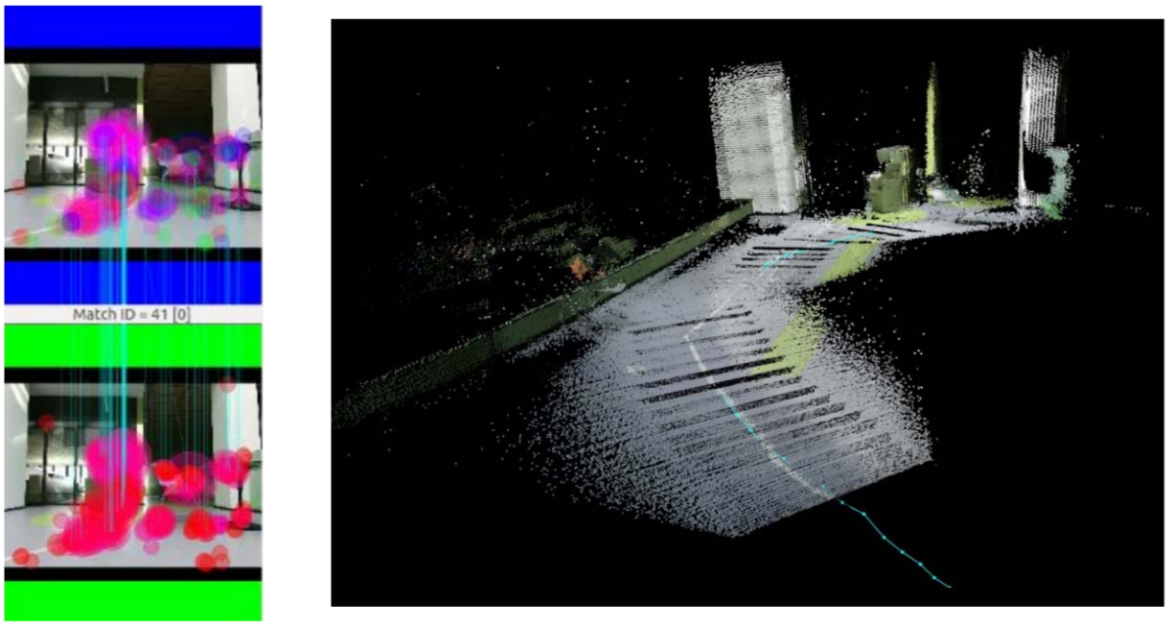


Figure 5.5: RTABMap loop closure detection (left) and point cloud dense map (right) [79].

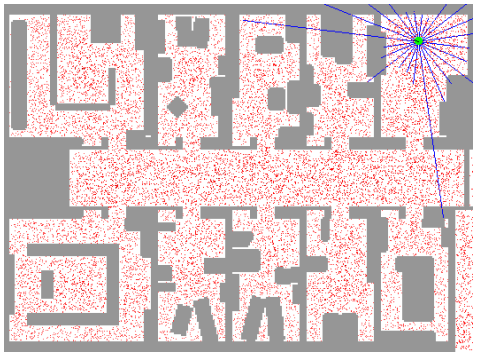
5.2 Navigation

Once the robotic agent is able to map its environment and has the means to plot a path given the start and finish points in said environment, to achieve full autonomy only needs to localise itself in the global environment map. This is made by applying algorithms which infer the pose from sensor data outputs. In general, the exact pose is not known, it is probabilistically estimated. This is the case of Markov localisation [80] and EKF localisation [81].
As Cen and Matsuhira state in [82], we can have three localisation problems. Position tracking, where the goal is to compensate for the odometry induced error, after the initial position

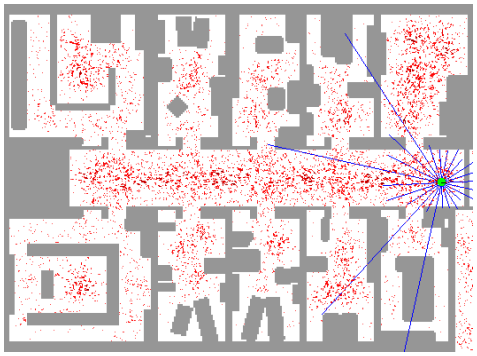
is known. A second, named global localisation, which aims to define the robot's pose by interpreting sensor readings. And the ultimate test to a localisation algorithm is the kidnapped problem that occurs when a well localised robot is suddenly teleported to another location without being told where.

5.2.1 AMCL

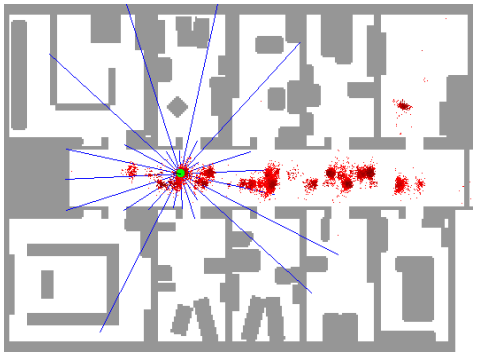
AMCL is a PF based probabilistic algorithm which aims to solve the localisation problem for a robot moving in 2-D [83]. It is also known as Kullback-Leibler Distance (KLD) sampling because it measures the approximation error by the Kullback-Leibler distance, and its fundamental idea is to restrict the approximation error introduced by the PF's sample-based representation. The major difference from traditional Monte Carlo Localization (MCL) [84] is the real time adaptiveness of the sample set size. Previous alternatives proposed approaches in which the sample size was variable [85], but never during state estimation *runtime*. This adaptive approach uses a larger number of samples for higher levels of uncertainty and a smaller one when uncertainty is related to a reduced space. Having this "on-the-fly" resizing drastically reduces the computational overhead. Figure 5.6 shows AMCL's progression, it can be seen that as new sensor readings are obtained, the location accuracy increases considerably.



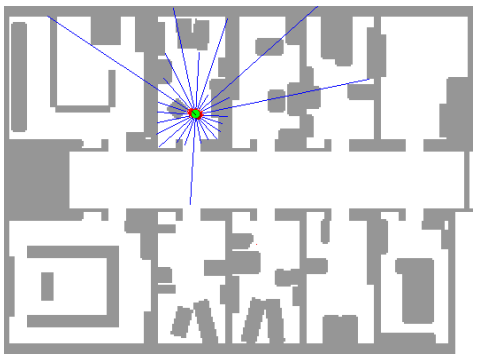
(a) First iteration with high uncertainty levels.



(b) With increasing number of sensor readings the belief increases.



(c) Uncertainty level is further reduced.



(d) Last iteration with minimal localisation uncertainty.

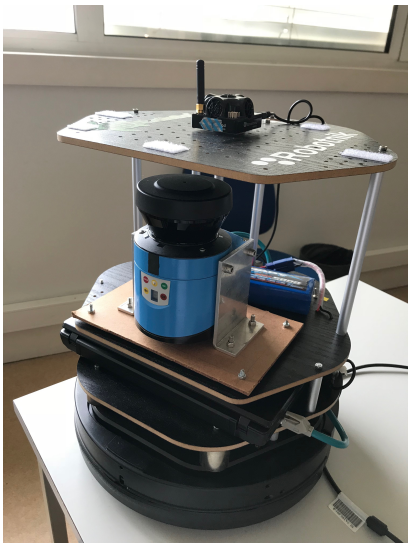
Figure 5.6: AMCL progress for pose estimation [86].

5.2.2 Active beacons

The supra referenced algorithms have a high success rate in resolving the position tracking problem and reasonable when it comes to the global position problem. Taking a long time, depending on map's size, and requiring considerable "blind" travelling distance and movement in order to determine the final approximate position for the global localisation problem. In situations where non-detectable obstacles exist, such as downward stairs, this delay can be hazardous leading to unavoidable collisions.
Therefore, the inclusion of an active localisation system like beacons can be a positive add-on reducing the uncertainty area to a much smaller size. If dealing with the kidnapped robot situation, the system can provide a new position input to the localisation algorithm as soon as it detects an abnormal position update.

5.3 Turtlebot 2

After a detailed analysis of the state-of-the-art out-of-the-box options for a medium size indoor robot, turtlebot 2 from Robotnik was the chosen one. Turtlebot 2 has the advantage of being a versatile robot in which the hardware components work on the *Plug & Play* principle.
We took an out of the box robot, with nothing but its *kobuki* mobile base and metal/wood structure. And plugged a LRF to work as the sensing component, a small computer which performs all the computation necessary and runs all major ROS nodes and a set of ultrasound beacons to be used as an indoor GPS system. The end result is presented in Figure 5.7a and Table 5.7b contains the brief HW specification.



(a) Turtlebot 2 after modifications.

Specs	
Base	Kobuki
Laser	Sick LMS100
Beacons	Marvelmind Robotics
laptop	Fujitsu LIFEBOOK P702

(b) General component specification.

Figure 5.7: Turtlebot 2 configuration overview.

5.4 Architecture

In order to act autonomously, three main problems must be solved: mapping, localisation and navigation. In the mapping phase, it obtains a map of the static part of the environment that enables self-localisation inside the building and, posteriorly, moving from point A to B in said map with minimal real time obstacle avoidance.
In order for a proper functioning robot, a few important ROS nodes must be up and running. First of all, the */mobile_base_nodelet_manager* starts the basic nodes for the robot, such as the *kobuki* node that handles all connections with the mobile base, publishes important data (e.g odometry, bumper and cliff sensor) and receives velocity input commands. Also, the */robot_state_publisher* node which uses the robot's joint angles, published by the previous referenced nodelet[1], as input and publishes the 3-D poses for each of the robot's links using a kinematic tree model. It publishes the robot state to *tf* (Figure 5.8) which maintains the relationship between coordinate frames over time.

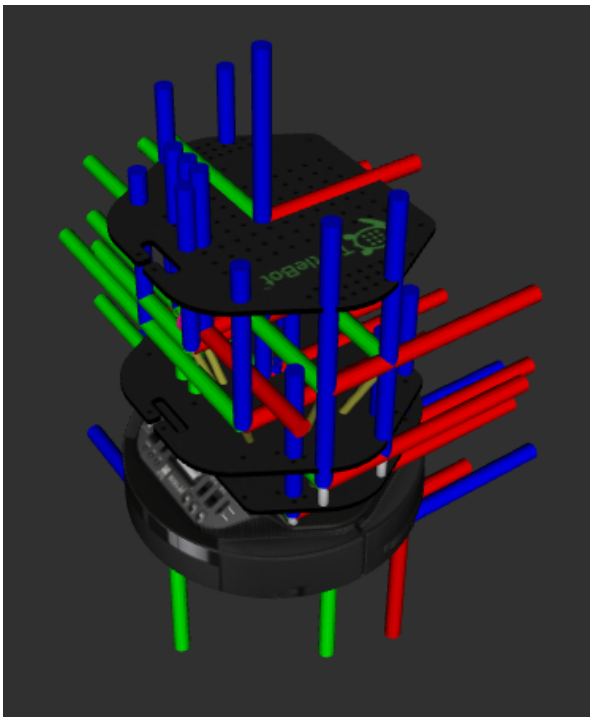


Figure 5.8: Turtlebot's 3-D *tf* as seen with *RVIZ*.

5.4.1 Mapping

To solve the mapping stage, GMapping was the chosen algorithm. It is relevant to reference that we use thirty (30) particles in the filter which is the default value. During tests it has proven to be the best value to balance the quality of mapping with computational load. In other words, for higher number of particles the map is slightly better but the load increases considerably.
As the environment contains stairways, in this phase the robot is controlled by teleoperation

[1] Way to run multiple algorithms in the same process.

in order to avoid obstacles not visible to the LRF. The LRF was used with the configuration from Table 5.1.

In this part, three ROS nodes play an important role. The */lms1xx* node is the driver for the used LRF and publishes the sensor readings to the */scan* topic. To teleoperate the robot, we use the */turtlebot_teleop_keyboard* node.

Finally, the */gmapping* node which outputs the 2-D occupancy grid map from the laser and pose data inputs.

The complete node interaction can be seen in the ROS graph from Figure 5.11.

Figure 5.9 represents the emergency floor plan for IEETA's entry floor in which the red delimited area represents the mapped section. Figure 5.10a represents the resulting map from IEETA's entry floor, obtained with GMapping. In this case, to solve the problem of unseen obstacles modifications were made to the map by adding black lines with an image editor. After these changes visible in Figure 5.10b, the robot no longer considers those places as free space, therefore does not include them in the possible path list.

Scanning frequency	50 Hz
Angular resolution	0.5°
Aperture angle	270°, [-45, 225]
Operating range	20 m

Table 5.1: LRF configuration at mapping stage.

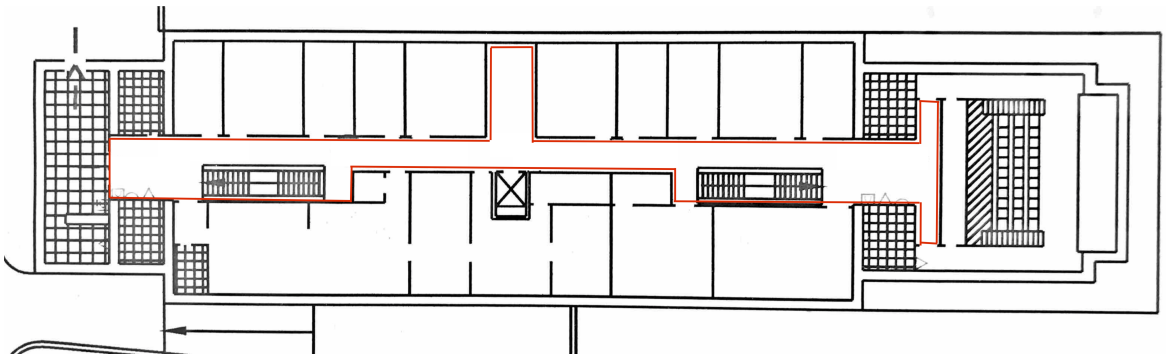
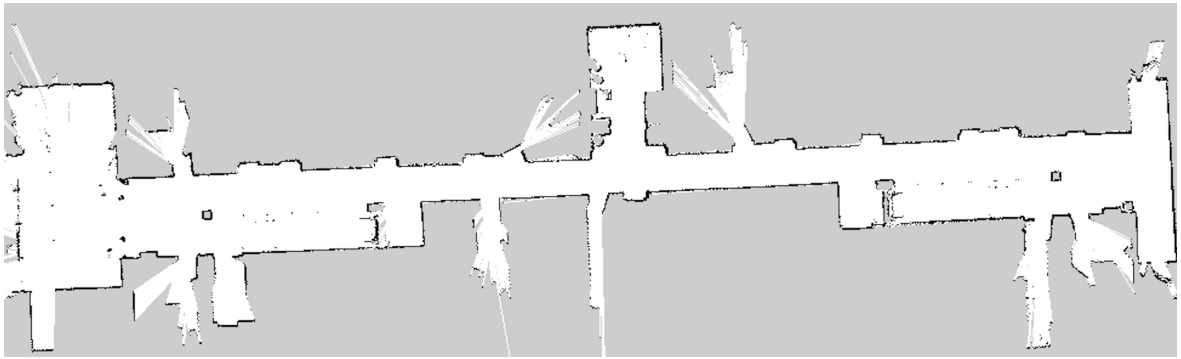
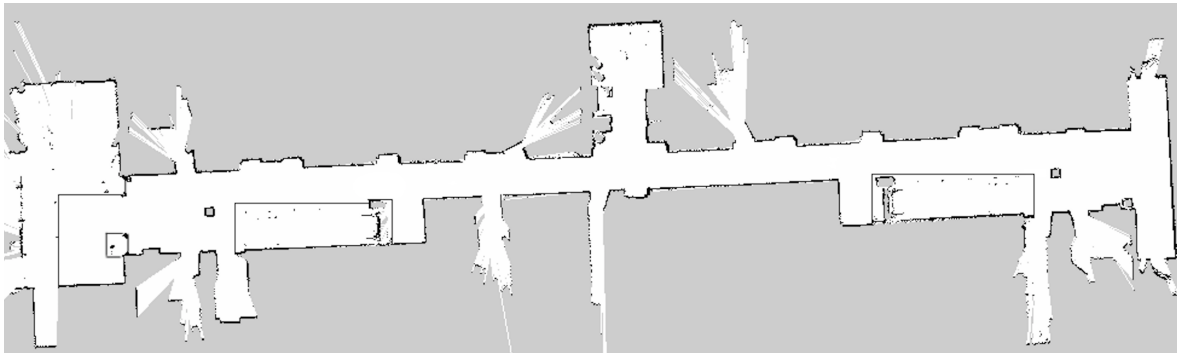


Figure 5.9: IEETA's floor number one (lobby) emergency floor plan. The red delimited area represents the mapped section.



(a) *Gmapping* result map.



(b) *Gmapping* result map after safety modifications.

Figure 5.10: Map from IEETA floor number one (lobby) obtained with *Gmapping*.

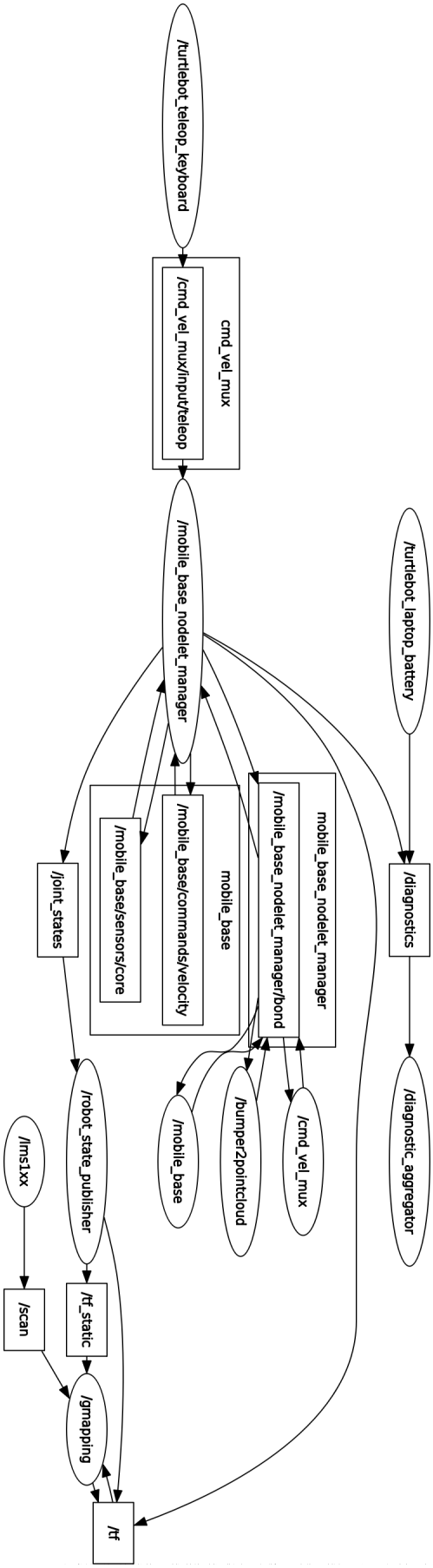


Figure 5.11: ROS generated graph showing all nodes and topics in the mapping stage.

5.4.2 Autonomous Navigation

After a proper mapping of the environment the agent must be able to perform autonomous task on it. To accomplish this we divided the problem in two parts: localisation and navigation.

Localisation

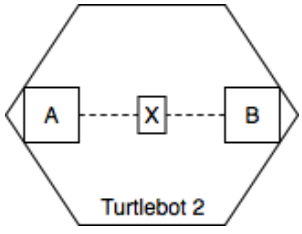
As referenced before, AMCL is a good algorithm to solve the localisation problem but in this specific environment (i.e containing unseen obstacles) letting the robot move freely until it find its location through sensor readings is not the best solution. To enhance and speed up this process two solutions emerged. The first in which the user could input the starting localisation and orientation of the robot using a graphical tool such as *RVIZ*. The second was to include in the environment an active tracking system that can provide the robot's position automatically.

To integrate the active system two possibilities were considered:

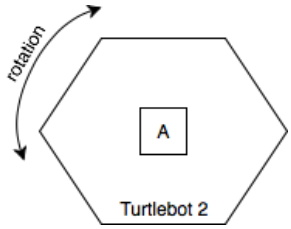
1. Use two mobile beacons on top of the robot configured as connected in the software in order to have an orientation. The robot's position was the middle point between beacons (Figure 5.12a).

2. Use only one mobile beacon to provide the position and create a rotation routine to determine the robot's orientation with sensor readings (Figure 5.12b).

With the two beacon configuration, when a small position variation occurs in any of the beacons, the angle changes drastically. This sudden angle variation causes a wrongful orientation input which leads to a malfunctioning AMCL.
Opposing this, the second option works better as long as the beacon systems provides meaningful input to the localisation algorithm. As soon as the systems has the robot's position (i.e pair x,y) provides it to AMCL with a dummy orientation and a covariance matrix with the rotation value representing said possible wrongful orientation input. After receiving that information, AMCL considers the received orientation as wrong and when the system performs the rotation routine (i.e 360 degrees each way) calculates the correct orientation through sensor readings. If the covariance matrix does not provide such information, AMCL will not make large orientation corrections.



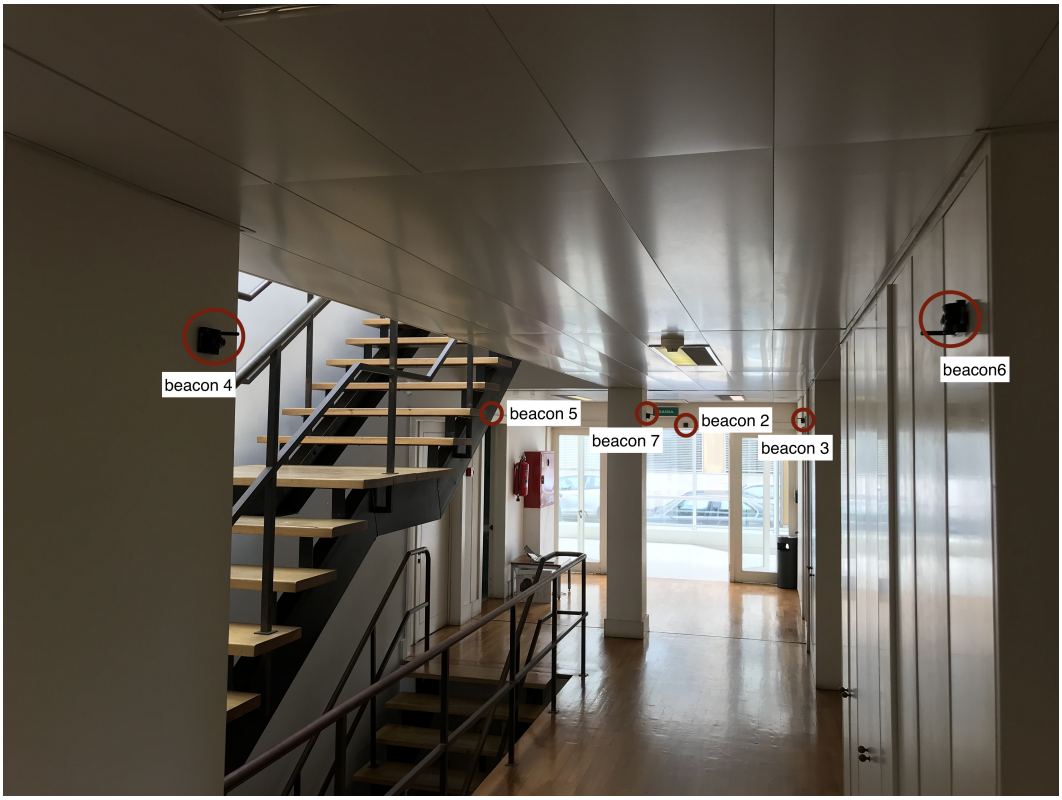
(a) Two beacons (A & B) with robot position being their middle point (X).



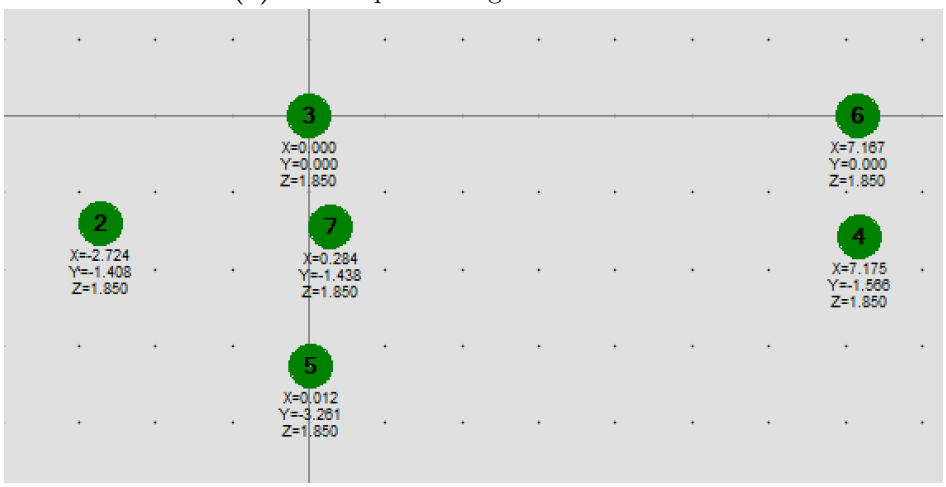
(b) Central beacon (A) with rotation routine.

Figure 5.12: Two possible beacon configuration to solve the orientation problem.

Figure 5.13 shows the active beacon system in place. The physical positioning is not mandatory. Beacon positioning can be arranged in any way, as long as the intended navigation area is covered. For this particular case, we only had eight beacons to work with. Due to the building's configuration (i.e presence of pillars, thick walls and many corners) we weren't able to completely cover the entry floor with the number of beacons at our disposal. Therefore, our focus went to the area near the docking station because it is the place where the system is more likely to be started, thus requiring a beacon position input to AMCL.



(a) Beacon positioning in the real world.



(b) Beacon positioning as seen in the proprietary software from Marvelmind[2].

Figure 5.13: Beacon positioning in IEETA's entry floor.

Having a working methodology for indoor localisation makes kidnapping detection possible. The developed solution to tackle this problem was a module which detects movements larger than a threshold distance value in the beacon system (Figure 5.14). This movements are only considered as kidnapping if the robot's motors are idle. Immediately upon determination, the new position is provided to AMCL and the rotation routine is performed so it can obtain the orientation.

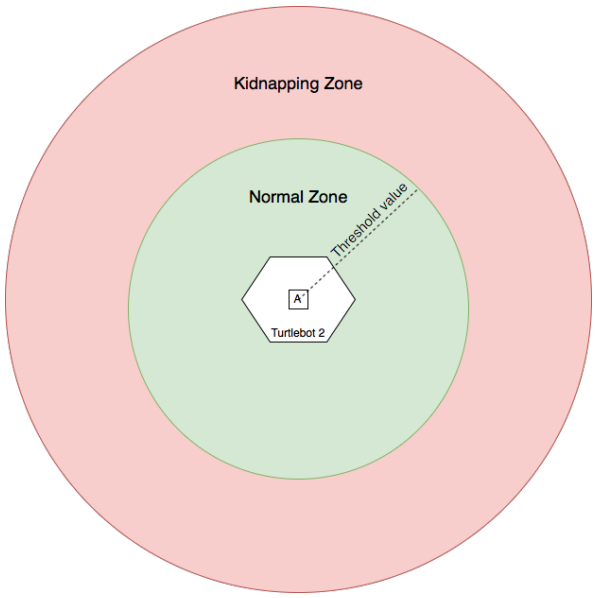


Figure 5.14: Scheme representing the kidnapping conditions.

Navigation

Once AMCL is correctly localised, the next step is to plot a course from the current position until the destination. *A** was the chosen algorithm to solve the shortest path problem due to its high computation speed. At this stage, real time obstacle detection is made as seen in Figure 5.16a and a path to the destination is computed (Figure 5.16b). This detection allows collision avoidance (Figure 5.16c) whenever a new obstacle emerges on the plotted path and path recalculation, if possible, in case that the path is no longer usable (Figure 5.16d).
In this part, besides the already described basic nodes and the */lms1xx* node, four other nodes are important to reference. Firstly, */amcl_node*, as the name suggests, corresponds to the AMCL's implementation to track the pose of a robot against a known map.
Secondly, */hedge_rcv_bin_node* is the driver node which serves as interface with the beacon system HW and publishes the mobile beacon's location to a specific topic.
Moreover, */turtlebot_subscriber_node* is responsible for providing the starting position to AMCL whenever it sees fit.
Ultimately, the */move_base* node that is an action implementation which, given a goal in the world, will attempt to reach it with a mobile base. It uses two planners, one local and one

[2] `https://marvelmind.com`

global with its respective costmaps, that are combined to accomplish navigation tasks. The detailed interaction of its components can be seen in Figure 5.15.
Also, the detailed node interaction for the autonomous navigation part is present in Figure 5.18.

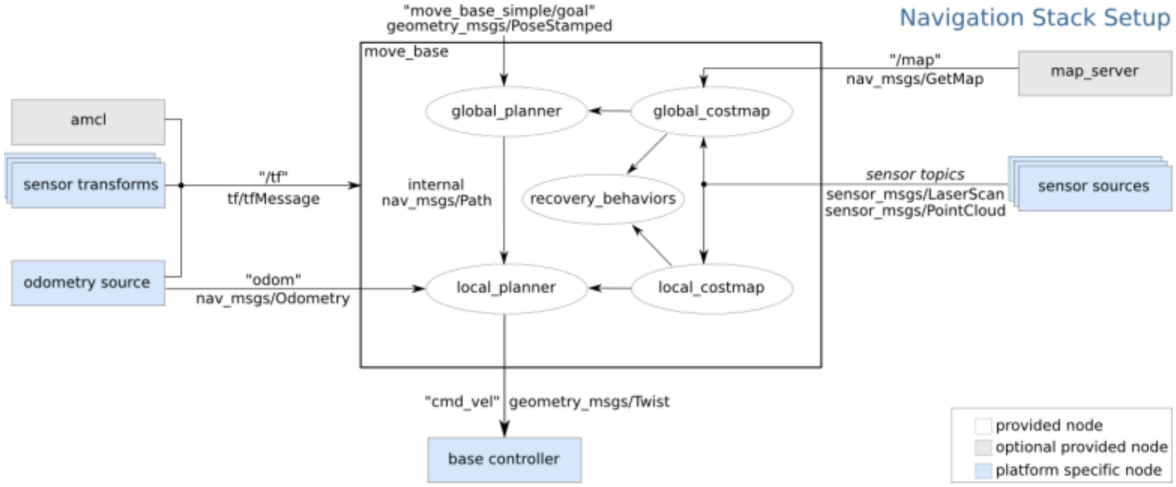
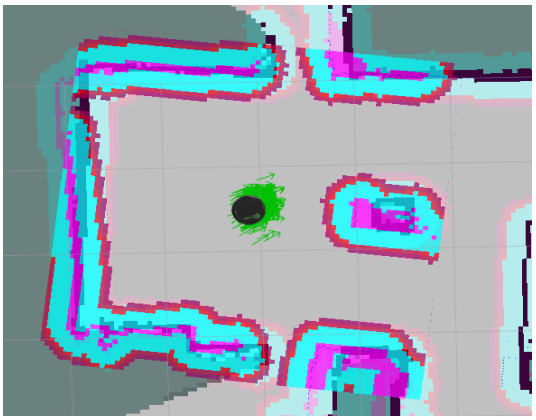
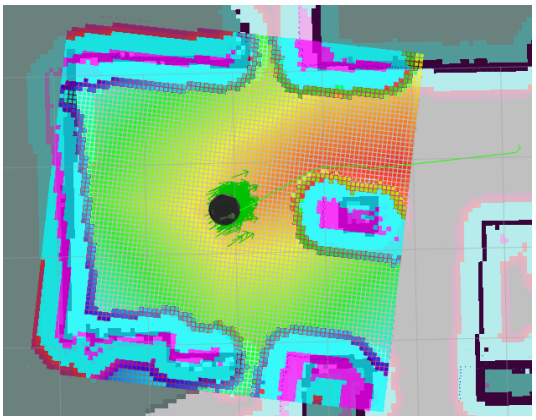


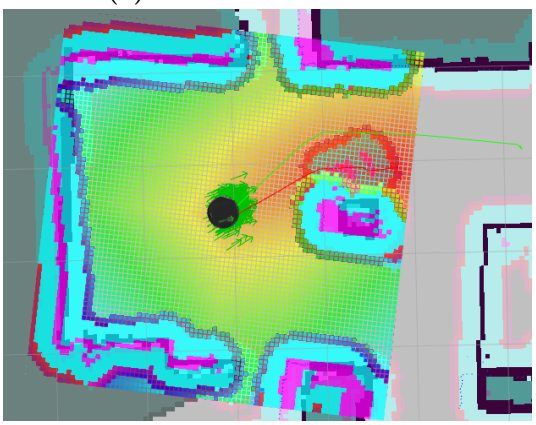
Figure 5.15: */move_base* detailed navigation stack interaction [87].



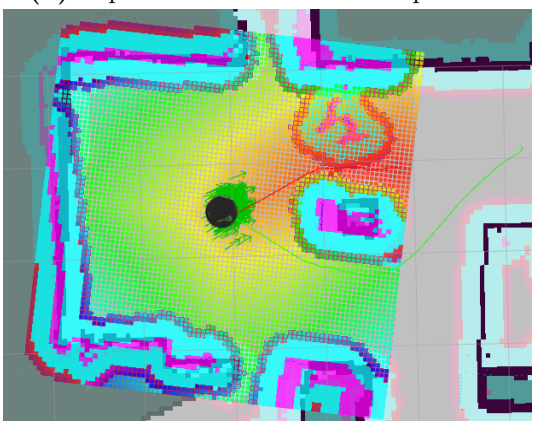
(a) Obstacles are detected.



(b) A path to the destination is plotted.



(c) When an object appears the path is adjusted (collision avoidance).



(d) A new path is computed, if the previous is no longer viable.

Figure 5.16: The four different stages of autonomous navigation.

The */move_base* node is responsible for the navigation task and is important to reference some configurations used to define the costmaps (Table 5.2):

- **inflation radius:** is the amount in meters by which obstacles are inflated, so that the robot can be considered as a point.

- **obstacle range:** is the maximum distance in meters that sensor readings are considered for real time obstacle detection.

- **XY goal tolerance:** tolerance in meters for the controller in the X & Y distance when achieving a goal.

- **Yaw goal tolerance:** tolerance in radians for the controller in yaw/rotation when achieving its goal.

- **recovery behaviours:** special routines to be executed in the event of an unmapped blocking obstacle is detected.

Inflation radius	0.3 m
obstacle range	2.5 m
XY goal tolerance	0.15 m
Yaw goal tolerance	0.3 rad
recovery behaviour 1	rotation
recovery behaviour 2	costmap reset

Table 5.2: Used values for the *move_base node*.

In situations where, after executing all recovery routines (Figure 5.17), there is no possible path to goal without collision the robot aborts and remains still until a new order is given. For this phase, the LRF's aperture angle was reduced to 220° ([-20, 200]) in order not to consider the robot's body as an obstacle.

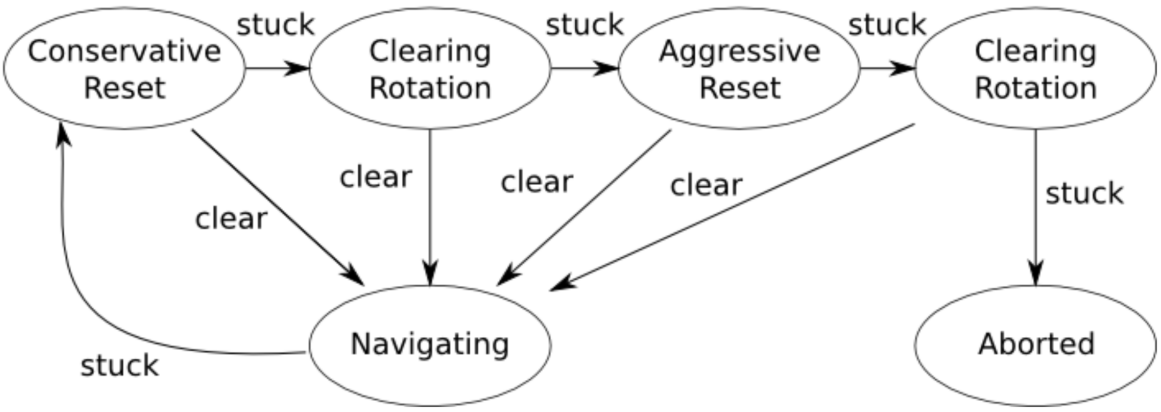


Figure 5.17: */move_base* recovery behaviour when the robot perceives itself as stuck [88].

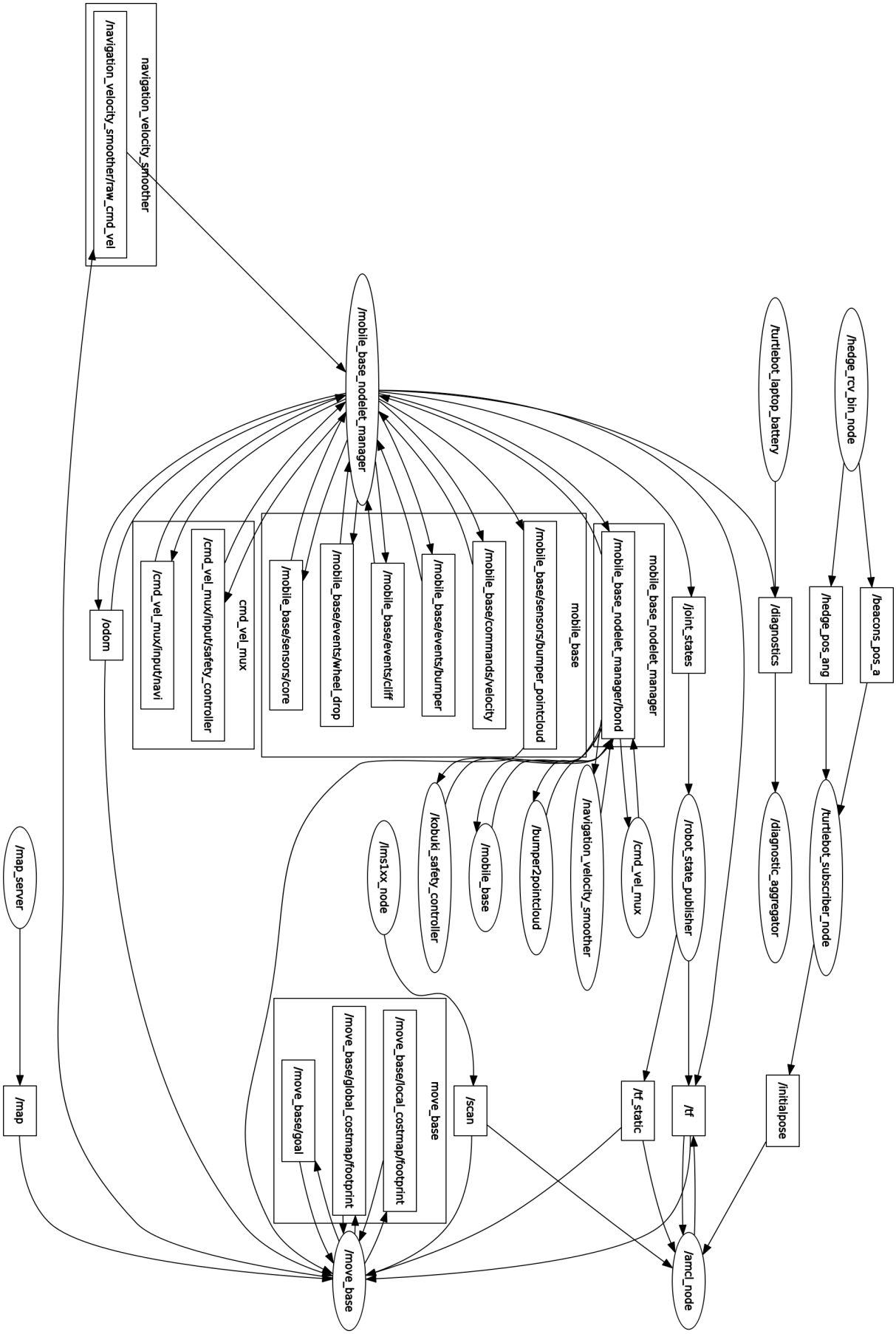


Figure 5.18: ROS generated graph showing all nodes and topics in the autonomous navigation stage.

5.5 User-System Interaction

To facilitate the user interaction with the system and enable operations such as teleoperation and the input of autonomous navigation goals, a GUI was created using python's open source library for application development *Kivy*[3]. It was developed to communicate with system over *SSH*[4], therefore not requiring heavy ROS installation. Also, another advantage of having the interface communicating over *SSH* is that if, for any reason, the connection between the *onboard* machine and the GUI is lost, the system stops.
Having a possible touch screen interaction in mind, this GUI, at launch time, presents the user with three main possibilities: teleoperation, autonomous navigation and auto-docking routine (represented by the dock icon). The opening window can be seen in Figure 5.19.

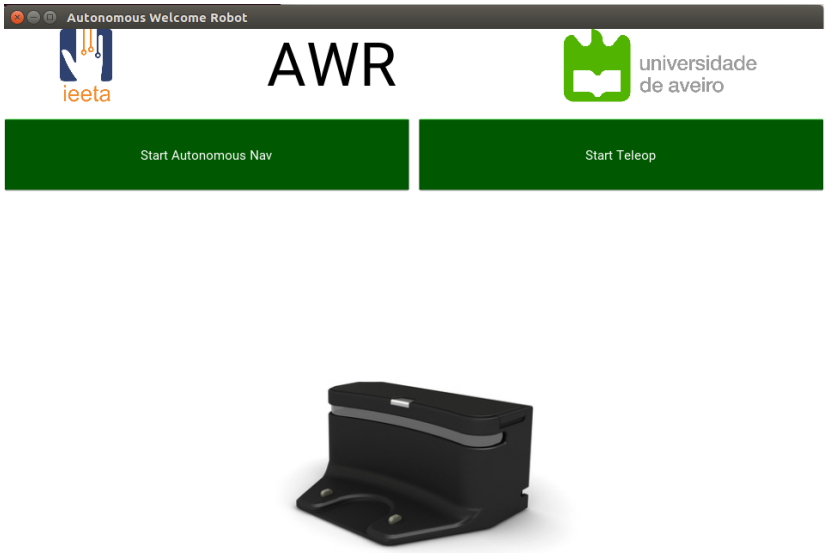


Figure 5.19: System GUI opening window.

5.5.1 Teleoperation

This interaction mode, represented in Figure 5.20, is very useful for situations in which small adjustments to the robot's position are required. Also, has proven to be a major feature at debug time. Once selected, an *SSH* session is opened with the robot's machine and the teleoperation node launched. Then, the movement buttons are converted into velocity commands for the robot and, when clicked, those commands are published to the */cmd_vel_mux* topic.

[3] `https://kivy.org`
[4] Protocol for operating network services securely: `https://www.ssh.com`

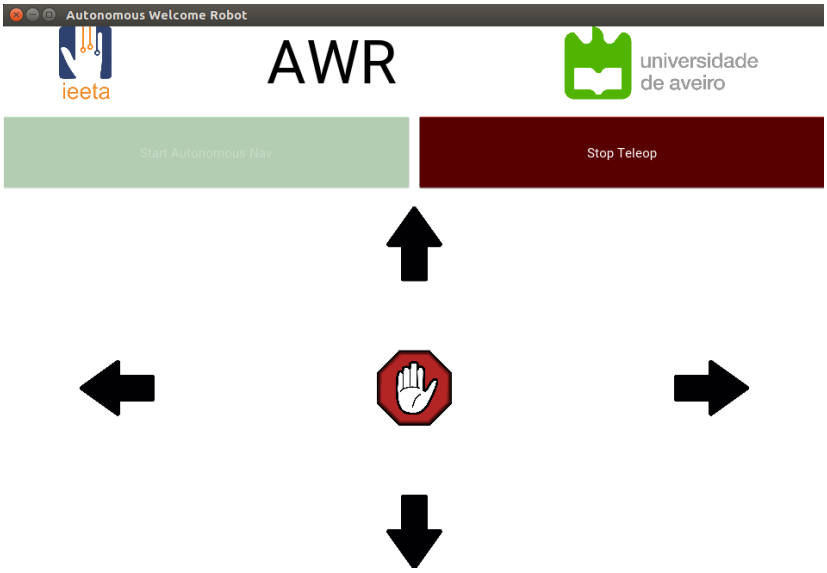


Figure 5.20: System GUI when in teleoperation mode.

5.5.2 Autonomous Navigation

Being the most import part, autonomous navigation interaction is made via set of buttons containing the possible destinations inside the building to where the robot can guide the user. The destinations are dynamically loaded from a dedicated file containing a list of destinations and its corresponding coordinates in the map. By doing this, we have an easily configurable solution where the user can add, remove or change target destinations directly in the text file without needing to change the GUI's source code. Once a destination is clicked, its corresponding coordinates are published to the */move_base/goal* topic over *SSH* and the robot starts its movement to the specified target.
To provide visual feedback, the GUI automatically splits the screen with *RVIZ*[5] which displays the live positioning of the robot in the map for basic users and also enables more complex interactions for expert users. Figure 5.21 illustrates this mode. At the end of each command the interface displays a notification with the end result status (Figure 5.22) and awaits for a new instruction (new destination or returning to dock).

[5]ROS visualisation tool: `http://wiki.ros.org/rviz`

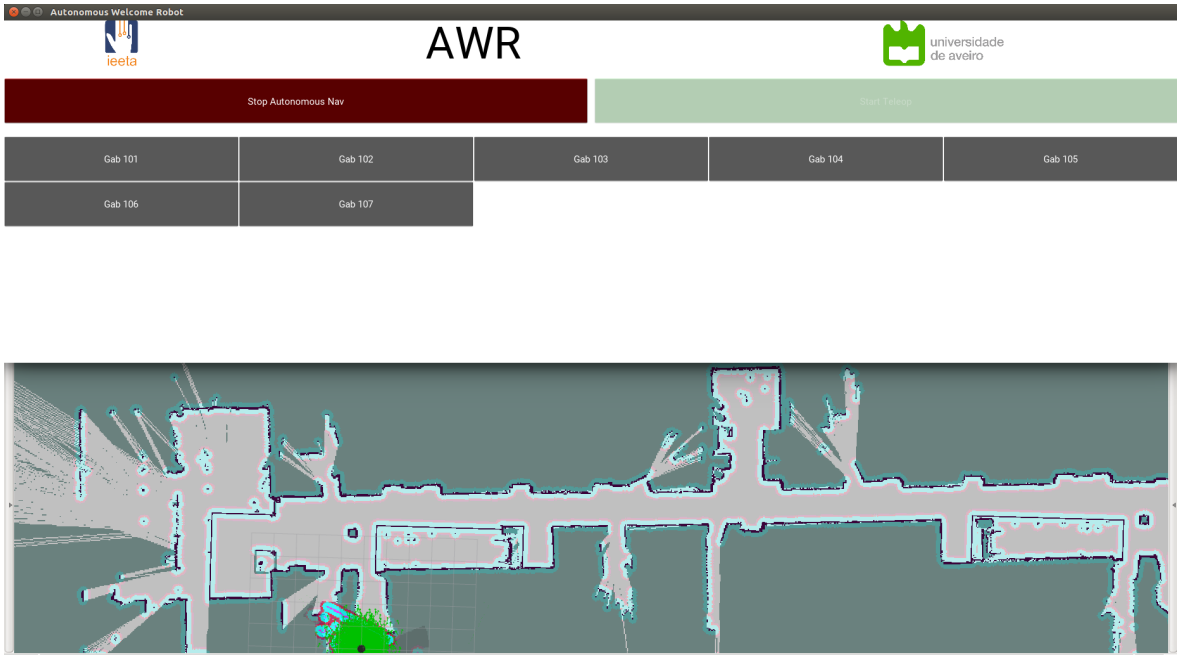


Figure 5.21: System GUI when autonomous navigation mode selected.

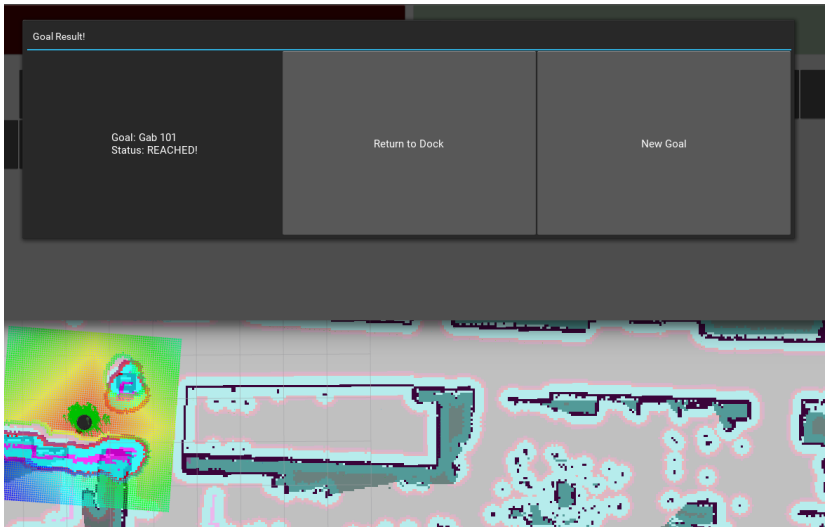


Figure 5.22: GUI feedback for destination command.

5.6 System Limitations

During the development of this case study, some problems were encountered. Some are more relevant than others and we did the best to mitigate them.

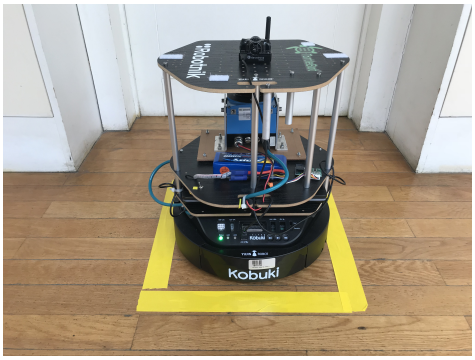
A major system limitation is the LRF positioning. As it is positioned at thirty centimetres from the ground it means that obstacles lower than said height will not be detected. If those objects are already present in the environment at mapping time they can be taken into account in the same way that stairways are. But if they only exist during navigation time, the robot, most certainly will hit them.

As it is an ultra-sound technology, the beacon system can sometimes suffer from interference. We estimate that this interference is caused by the high quantity of electronic devices present in the building. As a consequence, it transforms the beacon positioning into a more time consuming task.
Also, in order to have a proper functioning system, a correct matching between the beacon physical positioning and the map obtained by SLAM must be made. This is done by using measurement tools (e.g measuring tape) and providing those transformations to the beacon proprietary software. Even though static, this method induces an error which was taken into account when providing the covariance matrix to AMCL.
Although it has never failed, as a result of having all nodes running on the *onboard* machine, occasionally we could tell that it was struggling to keep up. An easy solution for this problem can be by adding a computer with more computational power.
Moreover, as the LRF is positioned in the front of the robot, the centre mass is also moved toward the front. This results in a, sometimes, malfunctioning auto-docking routine or in a dock dragging when decoupling from it. The solution ended up being the fixation of the docking station to the wall.

5.7 Results

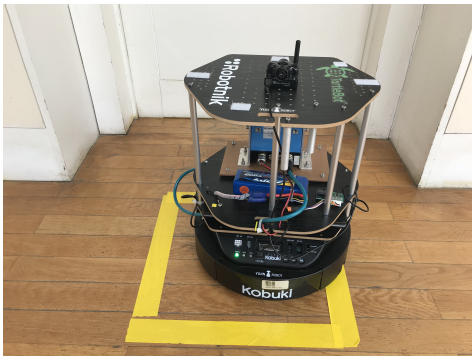
In order to test the overall system performance, we executed a series of twenty (20) destination orders to the same location, each having a different starting position. The robot was positioned in the intended destination and a square box was made with yellow tape around it as seen in Figure 5.23. Establishing this visual goal area enabled us to evaluate the system accuracy. Choosing this method to evaluate the system can be seen as a consequence of the overall system object of guiding a person into a location (e.g office), where the final numerical positioning of the robot has no significant impact. As long as the robot reaches the vicinity of the defined box, it is considered as an accomplished task. The objective was successfully reached in all of the twenty performed orders.
Figure 5.24 shows four visual results from the set of twenty performed, as an example.

(a) Top view.

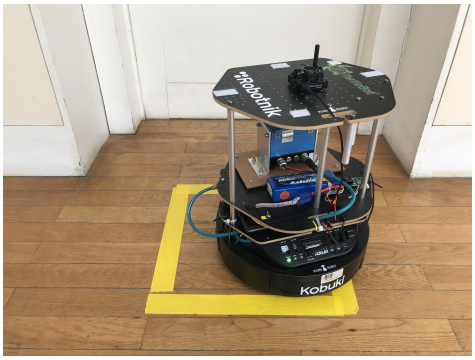


(b) Horizontal view.

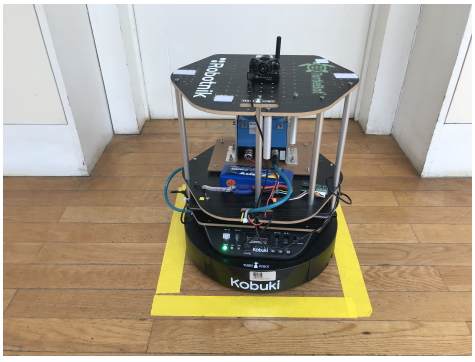
Figure 5.23: Destination box definition (yellow tape delimits the robot's footprint).



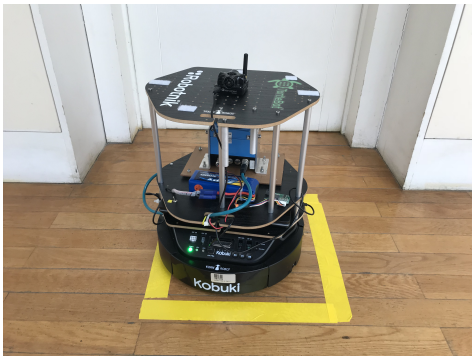
(a) First test result.



(b) Second test result.



(c) Eighteenth test result.



(d) Nineteenth test result.

Figure 5.24: Example of obtained practical results.

CHAPTER 6

Conclusion

In this thesis, we suggest a capable solution to solve the problem of autonomous navigation inside a building. We analysed and studied the problems and available solutions related to the three major parts of the proposed objective.
From the study performed we learned how the algorithms for mapping, path planning and navigation work. This established a knowledge base for the practical development.
Chapter 4 was the first practical execution in which we accomplished the proposed objective of having an agent capable of mapping a maze and posteriorly navigate on it autonomously. This can be considered as the starting point for the study of the active system's potentialities for indoor localisation. From the work done here, we concluded that said system is accurate and can be an improving factor for any indoor navigation system.
Chapter 5 is the major case study for this thesis in which the concepts studied before were combined. Analysing the end result of this chapter, we can say that it performs well for the proposed objectives and it presents a user-friendly methodology for environments with low level of mutation over time (i.e low variation of environment's physical characteristics). It is important to emphasise the added value of having an active beacon system to complement the global localisation algorithm (AMCL). This additional component, when compared with traditional solutions, strongly reduces the system complexity for the end user.
Overall, we can conclude that active systems such as the beacon system used in the practical part of this thesis, can be an enhancing factor for autonomous navigation system. Whether using it by itself as in the first case study, or as a complement to a sensor fusion solution for the second study, it has certainly proved to be a valid solution for the indoor localisation problem. Moreover, it can be the starting point for continuation solutions such as the ones proposed in the next Section (6.1).

6.1 Future work

After a retrospective reflection we concluded that more work could be done to enhance the obtained solution. In this section, we present succinct guidelines and ideas for possible future development.

- Develop Hardware (HW) and Software (SW) solutions to enable the detection of lower obstacles and floor depressions such as stairways.

- Study different possibilities for implementing an autonomous exploration and mapping algorithm.

- Development of a lightweight alternative for *RVIZ* in the autonomous navigation GUI.

- Study and develop techniques to reduce the probability of the robot aborting an order in case there's a blockage.

- Analyse methodologies to increase the active system reliability and diminish the medium interference.

- Study possible improvements of the kidnapped robot problem using the active beacon system.

- Implementing a human following system.

References

[1] G. Dudek and M. Jenkin, *Computational principles of mobile robotics*. Cambridge university press, 2010.

[2] *Mir100 @ Mir*. [Online]. Available: http://www.mobile-industrial-robots.com/media/1040/mir100.png?crop=0,0,0,0&cropmode=percentage&width=800&height=400&rnd=131598730240000000.

[3] *Pepper @ SoftBank*. [Online]. Available: http://infozene.com/blogging/wp-content/uploads/2017/09/Softbank-humanoid-pepper-robot-427x640.jpg.

[4] *Gita @ Piaggio*. [Online]. Available: https://cdn2.vox-cdn.com/uploads/chorus_asset/file/7933321/Gita.jpg.

[5] R. Siegwart, *Introduction to Autonomous Mobile Robots*, 9. 2011, vol. 53, p. 472, ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[6] *ASIMO @ Honda*. [Online]. Available: https://s.hswstatic.com/gif/asimo-1a.jpg.

[7] *ATLAS @ Boston Dynamics*. [Online]. Available: https://i.pinimg.com/originals/d2/70/48/d270481c1282585fc9ff17a4f14e515e.jpg.

[8] B. Dynamics, *AlphaDog*. [Online]. Available: https://o.aolcdn.com/images/dims?quality=100&image_uri=http%3A%2F%2Fwww.blogcdn.com%2Fwww.engadget.com%2Fmedia%2F2012%2F09%2Fboston-dynamics-alphadog-ls3-darpa-demo.jpg&client=cbc79c14efcebee57402&signature=e5bd2223459a26e6887643b24a31e34df7359df2.

[9] *NAO @ SoftBank*. [Online]. Available: https://www.robotlab.com/hubfs/images/Blog/Red-NAO---Top.png?t=1527893140150.

[10] H. R. Everett, *Sensors for mobile robots*. AK Peters/CRC Press, 1995.

[11] V Nguyen, A Martinelli, N Tomatis, and R Siegwart, "A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics", *International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[12] Sheng Fu, Hui-ying Liu, Lu-fang Gao, and Yu-xian Gai, "SLAM for mobile robots using laser range finder and monocular vision", *2007 14th International Conference on Mechatronics and Machine Vision in Practice*, no. 1, pp. 91–96, 2007. DOI: 10.1109/MMVIP.2007.4430722.

[13] Y. Kwon and J. Lee, "A stochastic map building method for mobile robot using 2-d laser range finder", *Autonomous Robots*, vol. 200, pp. 187–200, 1999.

[14] Hizook, *SICK LMS 100 Laser RangeFinder*. [Online]. Available: http://www.hizook.com/files/users/3/SICK_LMS100.jpg.

[15] R. Components, *TurtleBot 2 - ROS mobile robot*. [Online]. Available: https://www.roscomponents.com/26-thickbox_default/turtlebot-2.jpg.

[16] Turck, *Automated Guided Vehicle*. [Online]. Available: https://www.turck.us/static/img/VDL_Weweler_2_680x382-turck-image.jpg.

[17] *Service Robots @ IFR*. [Online]. Available: https://ifr.org/service-robots/.

[18] *AMIGO*. [Online]. Available: http://roboticopenplatform.org/wiki/images/thumb/9/90/Amigo_BIG.jpg/400px-Amigo_BIG.jpg.

[19] *ROOMBA*. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/2/27/ÐāÐ¿ÐšÐ¿ÑĆ_Ð¿ÑŃÐżÐµÑĄÐ¿ÑĄ_Roomba_780.jpg.

[20] *PATROLBOT*. [Online]. Available: http://commonhomerobots.weebly.com/uploads/3/8/0/0/38004467/6165618_orig.jpg.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System", *Icra*, vol. 3, no. Figure 1, p. 5, 2009, ISSN: 0165-022X. DOI: http://www.willowgarage.com/papers/ros-open-source-robot-operating-system. arXiv: 1106.4561.

[22] A. Koubaa, *Robot Operating System (ROS)*, Volume 2. 2017, vol. 707, ISBN: 978-3-319-54926-2. DOI: 10.1007/978-3-319-54927-9.

[23] *ROS_basic_concepts @ ros.org*. [Online]. Available: http://ros.org/images/wiki/ROS_basic_concepts.png.

[24] Z. Yang and Y. Liu, "Quality of trilateration: Confidence-based iterative localization", *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 631–640, 2010, ISSN: 10459219. DOI: 10.1109/TPDS.2009.90.

[25] A. Bekkelien, "Bluetooth indoor positioning", *Master's thesis, University ...*, no. March, p. 1, 2012.

[26] A. K. Hossain and W. S. Soh, "A comprehensive study of bluetooth signal parameters for localization", *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, 2007, ISSN: 2166-9570. DOI: 10.1109/PIMRC.2007.4394215.

[27] J. Bachrach and C. Taylor, "Localization in Sensor Networks", *Handbook of Sensor Networks: Algorithms and Architectures*, pp. 277–310, 2005, ISSN: 0387352694. DOI: 10.1002/047174414X.ch9.

[28] Widyawan, M. Klepal, and D. Pesch, "Influence of Predicted and Measured Fingerprint on the Accuracy of RSSI-based Indoor Location Systems", *2007 4th Workshop on Positioning, Navigation and Communication*, vol. 2007, pp. 145–151, 2007. DOI: 10.1109/WPNC.2007.353626.

[29] R. K. Moloo and V. K. Digumber, "Low-cost mobile GPS tracking solution", *Proceedings of the 2011 International Conference on Business Computing and Global Informatization, BCGIn 2011*, pp. 516–519, 2011. DOI: 10.1109/BCGIn.2011.136.

[30] R. Bajaj, S. Ranaweera, and D. Agrawal, "GPS: location-tracking technology", *Computer*, vol. 35, no. 4, pp. 92–94, 2002, ISSN: 0018-9162. DOI: 10.1109/MC.2002.993780.

[31] *GPS24goldenSML @ upload.wikimedia.org*. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/2/27/GPS24goldenSML.gif.

[32] *Image-7 @ i1.wp.com*. [Online]. Available: https://i1.wp.com/grindgis.com/wp-content/uploads/2017/06/image-7.png.

[33] *Rede SERVIR*. [Online]. Available: http://1.bp.blogspot.com/_ZR6RZXuEc4Q/SSlsnwFqPdI/AAAAAAAAAOg/3-AB8wvUnPo/s1600/rede_servir.jpg.

[34] *ReNEP*. [Online]. Available: https://www.researchgate.net/profile/Rui_Juliao/publication/290325132/figure/fig3/AS:613872390123530@1523369991121/Figura-4-Rede-Nacional-de-Estacoes-Permanentes-RENEP.jpg.

[35] V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara, "Accurate GSM Indoor Localization", no. iv, pp. 141–158, 2005, ISSN: 15741192. DOI: 10.1007/11551201_9.

[36] A. Varshavsky, E. de Lara, J. Hightower, A. LaMarca, and V. Otsason, "GSM indoor localization", *Pervasive and Mobile Computing*, vol. 3, no. 6, pp. 698–720, 2007, ISSN: 15741192. DOI: 10.1016/j.pmcj.2007.07.004.

[37] B. Denby, Y. Oussar, I. Ahriz, and G. Dreyfus, "High-performance indoor localization with full-band GSM fingerprints", *Proceedings - 2009 IEEE International Conference on Communications Workshops, ICC 2009*, pp. 0–4, 2009. DOI: 10.1109/ICCW.2009.5207991.

[38] *Cell-phone-gps-gsm-location-3 @ easytechnow.com*. [Online]. Available: https://easytechnow.com/wp-content/uploads/2016/05/cell-phone-gps-gsm-location-3.jpg.

[39] C. H. Lim, Y. Wan, B. P. Ng, and C. M. S. See, "A real-time indoor WiFi localization system utilizing smart antennas", *IEEE Transactions on Consumer Electronics*, vol. 53, no. 2, pp. 618–622, 2007, ISSN: 00983063. DOI: 10.1109/TCE.2007.381737.

[40] S. Bellofiore, J. Foutz, C. a. Balanis, and A. Spanias, "Smart Antennas for Wireless Communications", no. 1, pp. 26–29,

[41] J. Winters, "Smart antennas for wireless systems", *IEEE Personal Communications*, vol. 5, no. 1, pp. 23–27, 1998, ISSN: 10709916. DOI: 10.1109/98.656155.

[42] M. Ciurana, F. Barceló, and S. Cugno, "Indoor tracking in WLAN location with TOA measurements", *Proceedings of the international workshop on Mobility management and wireless access - MobiWac '06*, p. 121, 2006. DOI: 10.1145/1164783.1164806.

[43] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems 1", *Journal of Fluids Engineering*, vol. 82, no. Series D, pp. 35–45, 1960, ISSN: 00219223. DOI: 10.1115/1.3662552. arXiv: NIHMS150003.

[44] E. Mok and G. Retscher, "Location determination using wifi fingerprinting versus wifi trilateration", *Journal of Location Based Services*, vol. 1, no. 2, pp. 145–159, 2007, ISSN: 17489733. DOI: 10.1080/17489720701781905.

[45] A. W. S. Au, C. Feng, S. Valaee, S. Reyes, S. Sorour, S. N. Markowitz, D. Gold, K. Gordon, and M. Eizenman, "Indoor tracking and navigation using received signal strength and compressive sensing on a mobile device", *IEEE Transactions on Mobile Computing*, vol. 12, no. 10, pp. 2050–2062, 2013, ISSN: 15361233. DOI: 10.1109/TMC.2012.175.

[46] A. Ens, F. Höflinger, J. Wendeberg, J. Hoppe, R. Zhang, A. Bannoura, L. M. Reindl, and C. Schindelhauer, "Acoustic self-calibrating system for indoor smart phone tracking", *International Journal of Navigation and Observation*, vol. 2015, 2015, ISSN: 16876008. DOI: 10.1155/2015/694695.

[47] W. Mao, Z. Zhang, L. Qiu, J. He, Y. Cui, and S. Yun, "Indoor Follow Me Drone", in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '17*, 2017, pp. 345–358, ISBN: 9781450349284. DOI: 10.1145/3081333.3081362.

[48] S. B. Kim, I. O. Lee, D. I. Cho, and J. M. Lee, *Dynamic localization of a mobile robot with active beacon sensors*, PART 1. IFAC, 2006, vol. 4, pp. 921–925, ISBN: 9783902661173. DOI: 10.3182/20060912-3-DE-2911.00158.

[49] M McCarthy and H. Muller, "Positioning with independent ultrasonic beacons", *Dept. Comput. Science, Univ. Bristol, Bristol, ...*, 2005.

[50] M. Robotics, "Marvelmind Indoor Navigation System Operating Manual",

[51] Z. Chen, H. Zou, H. Jiang, Q. Zhu, Y. C. Soh, and L. Xie, "Fusion of WiFi, smartphone sensors and landmarks using the kalman filter for indoor localization", *Sensors (Switzerland)*, vol. 15, no. 1, pp. 715–732, 2015, ISSN: 14248220. DOI: 10.3390/s150100715. arXiv: 1007.0085.

[52] R. Zhang, A. Bannoura, F. Hoflinger, L. M. Reindl, and C. Schindelhauer, "Indoor localization using a smart phone", *IEEE Sensors Applications Symposium Proceedings*, pp. 38–42, 2013. DOI: 10.1109/SAS.2013.6493553.

[53] RMI-Teachers, "CiberRato Competition and Tools Robotic Competitions Micro-Rato Demos Installing the tools", pp. 1–39, 2015.

[54] ——, "File from-Robótica Móvel e Inteligente", pp. 2011–2012, 2012.

[55] R. Nambiar and M. Poess, *Performance Characterization and Benchmarking. Traditional to Big Data: 6th TPC Technology Conference, TPCTC 2014, Hangzhou, China, September 1–5, 2014. Revised Selected Papers*. Springer, 2015, vol. 8904.

[56] *512px-Shortest_path_with_direct_weights @ upload.wikimedia.org*. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/3/3b/Shortest_path_with_direct_weights.svg/512px-Shortest_path_with_direct_weights.svg.png.

[57] *Astar_progress_animation @ upload.wikimedia.org*. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/5/5d/Astar_progress_animation.gif.

[58] M. W. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem", *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001, ISSN: 1042296X. DOI: 10.1109/70.938381. arXiv: NIHMS150003.

[59] G. Q. Huang, A. B. Rad, and Y. K. Wong, "Online SLAM in dynamic environments", *2005 International Conference on Advanced Robotics, ICAR '05, Proceedings*, vol. 2005, pp. 262–267, 2005. DOI: 10.1109/ICAR.2005.1507422.

[60] N. Keivan and G. Sibley, "Online SLAM with any-time self-calibration and automatic change detection", *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 5775–5782, 2015, ISSN: 10504729. DOI: 10.1109/ICRA.2015.7140008. arXiv: 1411.1372.

[61] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems", *IEEE International Conference on Intelligent Robots and Systems*, pp. 573–580, 2012, ISSN: 21530858. DOI: 10.1109/IROS.2012.6385773.

[62] W Burgard, C Stachniss, G Grisetti, B Steder, R Kümmerle, C Dornhege, M Ruhnke, A. Kleiner, J. D. Tardós, W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, and M. Ruhnke, "A Comparison of SLAM Algorithms Based on a Graph of Relations A Comparison of SLAM Algorithms Based on a Graph of Relations Trajectory-based Comparison of SLAM Algorithms", pp. 2089–2095, 2009. DOI: 10.1109/IROS.2009.5354691.

[63] R. Ouellette and K. Hirasawa, "A comparison of SLAM implementations for indoor mobile robots", *IEEE International Conference on Intelligent Robots and Systems*, pp. 1479–1484, 2007. DOI: 10.1109/IROS.2007.4399575.

[64] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System", *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*, 2013. DOI: 10.1109/SSRR.2013.6719348.

[65] D. Pagac, E. M. Nebot, and H. Durrant-Whyte, "An Evidential Approach to Map-Bulding for Autonomous Vehicles", *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 623–629, 1998, ISSN: 1042296X. DOI: 10.1109/70.704234.

[66] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[67] *KF*. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/a/a5/Basic_concept_of_Kalman_filtering.svg/1701px-Basic_concept_of_Kalman_filtering.svg.png.

[68] L. Jetto, S. Longhi, and G. Venturini, "Development and experimental validation of an adaptive extended\nKalman filter for the localization of mobile robots", *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 219–229, 1999, ISSN: 1042-296X. DOI: 10.1109/70.760343.

[69] E. Wan and R Van Der Merwe, "The unscented Kalman filter for nonlinear estimation", *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pp. 153–158, 2002, ISSN: 15270297. DOI: 10.1109/ASSPCC.2000.882463.

[70] I. M. Rekleitis, "A particle filter tutorial for mobile robot localization", *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004.

[71] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM", *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010, ISSN: 1939-1390. DOI: 10.1109/MITS.2010.939925.

[72] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures", *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006, ISSN: 02783649. DOI: 10.1177/0278364906065387.

[73] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellised Particle Filters", *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, 2007, ISSN: 1552-3098.

[74] ——, *OpenSLAM.org*. [Online]. Available: `https://openslam-org.github.io/gmapping.html` (visited on 05/21/2018).

[75] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam", in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 1271–1278.

[76] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation", in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, IEEE, 2011, pp. 155–160.

[77] A. Nüchter, M Bleier, J. Schauer, and P. Janotta, "Improving google's cartographer 3d mapping by continuous-time slam", *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 543, 2017.

[78] M. Labbé and F. Michaud, "Memory management for real-time appearance-based loop closure detection", in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, IEEE, 2011, pp. 1271–1276.

[79] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ros-based visual slam methods in homogeneous indoor environment", in *Positioning, Navigation and Communications (WPNC), 2017 14th Workshop on*, IEEE, 2017, pp. 1–6.

[80] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments", *Journal of artificial intelligence research*, vol. 11, pp. 391–427, 1999.

[81] L. Chen, H. Hu, and K. McDonald-Maier, "Ekf based mobile robot localization", in *Emerging Security Technologies (EST), 2012 Third International Conference on*, IEEE, 2012, pp. 149–154.

[82] G. Cen and N. Matsuhira, "Mobile robot global localization using particle filters", *... and Systems, 2008. ...*, pp. 710–713, 2008. DOI: `10.1109/ICCAS.2008.4694593`.

[83] D. Fox, "KLD-sampling: Adaptive particle filters", *Advances in Neural Information Processing Systems 14*, vol. 14, no. 1, pp. 713–720, 2002, ISSN: 10495258. DOI: `10.1.1.21.5786`.

[84] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots", *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001, ISSN: 00043702. DOI: `10.1016/S0004-3702(01)00069-8`.

[85] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots", *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, no. May, pp. 1322–1328, 1999, ISSN: 1050-4729. DOI: `10.1109/ROBOT.1999.772544`.

[86] D. Fox, *Dieter Fox's MCL Animations*. [Online]. Available: `https://rse-lab.cs.washington.edu/projects/mcl/animations/global-floor.gif`.

[87] *Move Base*. [Online]. Available: `http://wiki.ros.org/move_base?action=AttachFile&do=get&target=overview_tf_small.png`.

[88] *Move Base Recovery Behaviour*. [Online]. Available: `http://wiki.ros.org/move_base?action=AttachFile&do=get&target=recovery_behaviors.png`.